

# Quiz<sup>?</sup> CREATOR

4. semesterprojekt  
Rapport, Gruppe 3  
28/5 2015





# Underskrifter

---

Dato for aflevering: **28/05/2015**

Antal normalsider for rapport: 24,6 (af 2400 tegn)

Gruppemedlemmer:

---

Anders Bæk Møller	201270727
-------------------	-----------

---

Bjørn Sørensen	201270432
----------------	-----------

---

Jesper Christensen	201270478
--------------------	-----------

---

Lasse Beck Thostrup	201271288
---------------------	-----------

---

Lars Harup Holm	201370744
-----------------	-----------

---

Loc Dai Le	201371016
------------	-----------

---

Michael Toft Jensen	201371180
---------------------	-----------

---

Vejleder: Frank Bodholdt Jakobsen



# Arbejdsfordeling

Områder	Anders	Bjørn	Jesper	Lasse	Lars	Loc	Michael
Kravspecifikation	✓	✓	✓	✓	✓	✓	✓
Use case beskrivelser	✓	✓	✓	✓	✓	✓	✓
Gui skitser	✓		✓	✓	✓		
Forundersøgelser							
GUI		✓					
Database							✓
Systemarkitektur							
Domænemodel						✓	
4+1		✓	✓	✓	✓		
Design							
Sekvensdiagrammer		✓					✓
Implementering og test							
AnswerQuiz	✓	✓		✓			
Group			✓				
ConfigGroup			✓				✓
CreateQuiz				✓	✓		
ConfigQuizzes		✓		✓	✓		
Search	✓					✓	
CreateUser							✓
ConfigUser						✓	
Login			✓				
FacebookRegister			✓				
Favorites					✓		
Validation		✓	✓				
Quiz-Group Relation popover		✓					
DAL(iteration 1)			✓				
DALEF		✓	✓	✓		✓	✓
QuizModel		✓	✓		✓	✓	
DoxyGen				✓			
Accepttest-specifikation							
Accepttest	✓	✓	✓	✓	✓	✓	✓



# Abstract

---

This report contains a thorough walkthrough of our fourth term project named Quiz Creator. The application does exactly what the name implies: it allows users to create and share quizzes among each other. It is an application where the possibilities are practically endless - whether a teacher wants to create quizzes for his or her students on a certain subject or if students create quizzes for each other to practice before a difficult examination, it is possible with Quiz Creator.

Work on this project began in early February. From the beginning the team decided to take advantage of the Scrum framework - most of the team members had worked in agile settings before, so it was an obvious choice. And in many aspects the project has been driven forward in an iterative way and this means that the project has been split up in different iterations, where each iteration has added more and more functionality.

Towards the end of the project most of the essential features found in the use cases have been added to the application. This report will give the reader an overview of how we achieved this result.





# Resume

---

Denne rapport forsøger at give et detaljeret indblik i arbejdet med vores 4. semesters projekt, som omhandler applikationen Quiz Creator. Quiz Creator er hverken mere eller mindre end hvad navnet antyder: en applikation, som gør brugeren i stand til at oprette og samtidig besvare en lang række quizzes som er blevet oprettet af andre brugere. Et sådant program har mange muligheder, bl.a. til undervisningen i gymnasiet, på universitetet eller i folkeskolen, hvor læreren kan udnytte programmet til at lave quizzes til eleverne for f.eks. at kontrollere at pensum er indlært.

Projektarbejdet begyndte i starten af februar og er sidenhen løbet parallelt med resten af studiet. Gruppen besluttede fra starten at følge Scrum-frameworket således at arbejdet blev delt op i sprints. Generelt har gruppen taget en agil tilgang til arbejdet, og således er arbejdet med Quiz Creator skredet frem i iterationer som efterhånden har tilføjet flere og flere funktionaliteter.

Ved projektets afslutning er de fleste væsentlige funktionaliteter implementeret i applikationen som foreskrevet af use case-beskrivelserne. Rapporten vil give et overblik over disse funktionaliteter og hvordan vi er nået frem til dem.



# Indholdsfortegnelse

---

<b>Kapitel 1</b>	<b>Ordliste</b>	<b>11</b>
<b>Kapitel 2</b>	<b>Indledning</b>	<b>13</b>
<b>Kapitel 3</b>	<b>Projektafgrænsning</b>	<b>15</b>
<b>Kapitel 4</b>	<b>Identificering af problemdomæner</b>	<b>17</b>
<b>Kapitel 5</b>	<b>Iteration 1</b>	<b>19</b>
5.1	Krav . . . . .	19
5.2	Forundersøgelse . . . . .	20
5.3	Grafisk brugergrænseflade design . . . . .	21
5.4	Arkitektur . . . . .	23
5.5	Database . . . . .	23
5.6	Dokumentationsværktøjet Doxygen . . . . .	24
5.7	Sprints . . . . .	24
5.8	Test . . . . .	25
<b>Kapitel 6</b>	<b>Iteration 2</b>	<b>27</b>
6.1	Krav . . . . .	27
6.2	Ikke-funktionelle krav . . . . .	28
6.3	Config og Create Quiz . . . . .	28
6.4	Answer Quiz . . . . .	29
6.5	Database . . . . .	30
6.6	Users . . . . .	31
6.7	Login system . . . . .	32
6.8	Sprints . . . . .	34
6.9	Test . . . . .	35
<b>Kapitel 7</b>	<b>Iteration 3</b>	<b>37</b>
7.1	Krav . . . . .	37
7.2	Answer Quiz . . . . .	38
7.3	Groups . . . . .	39
7.4	Favorites . . . . .	40
7.5	Azure web- og database-server . . . . .	40
7.6	Sprints . . . . .	40
7.7	Test . . . . .	41
<b>Kapitel 8</b>	<b>Projektforsløb og opfølgning</b>	<b>43</b>
8.1	Arbejdsproces . . . . .	43
8.1.1	Udviklingsmodeller . . . . .	43
8.1.2	Møder og referater . . . . .	44

8.1.3 Arbejdsfordeling . . . . .	45
8.2 Udviklingsværktøjer . . . . .	45
8.3 Fremtidigt arbejde . . . . .	46
8.4 Resultater . . . . .	47
8.5 Erfaringer . . . . .	48
<b>Kapitel 9 Konklusion</b>	<b>51</b>
9.1 Samlet konklusion . . . . .	51
9.2 Individuelle konklusioner . . . . .	51
<b>Kapitel 10 Bilags-CD indhold</b>	<b>57</b>
<b>Litteratur</b>	<b>59</b>

**AASH** Antal af samtidige hændelser (Use Case)

**ASE** Aarhus School of Engineering

**DAL** Data Access Layer

**EF** Entity Framework. Et .NET-framework til databasehåndtering

**GUI** Graphical User Interface

**UC** Use Case

**Session** En aktiv forbindelse mellem brugerens webbrowser og applikationen QuizCreator.

Tilgås applikationen fra endnu en fane, har brugeren altså 2 sessioner åbne. (Use Case)



# 2

## Indledning

---

Quizzer bruges i dagligdagen til alverdens ting. De benyttes i alt lige fra Trivial Pursuit til seriøse eksamener på gymnasiet eller på universitetet. Quizzer er alsidige - de kan være så korte eller lange som man ønsker, de kan indeholde spørgsmål om alt lige fra hvilken dag Danmark blev invaderet i 2. verdenskrig til hvilken mel-type der er bedst at bruge i fastelavnsboller (\*hvedemel er svaret).

Quizzer befinder sig overalt i dagligdagen og dog er der stadig ikke blevet skabt en solid platform, hvor brugere kan oprette og dele quizzer med hinanden. Med Quiz Creator vil vi tage de første spæde skridt i retning mod denne vision: Projektet skal i den første version opfylde en række use cases, herunder bl.a. Oprettelse af quiz, besvarelse af quiz, søgning af quiz, oprettelse af bruger, log in, log ud, oprette grupper til at sortere quizzer indenfor et bestemt område osv. En applikation af denne type er fleksibel nok til ikke kun at høre hjemme på én platform, og således vil det give mening både at lave en version til PC'er, Smartphones og tablets. Grundet den begrænsede tid vi har til projektet vil vi dog kun fokusere på PC-versionen som skal være en internet-applikation.

Denne rapport beskriver udviklingsforløbet igennem projektet. Eftersom der er blevet arbejdet agilt med Scrum i sprints, har vi valgt at opdele rapporten i iterationer således at læseren får nemt overblik over hvordan arbejdet er skredet frem, og hvilke milepæle der er sat hvornår.





# 3

## Projektafgrænsning

---

Fra starten af projektet lagde vi nogle klare afgrænsninger for hvor meget vi gerne ville have implementeret i webapplikationen. Det var vigtigt at få lagt nogle realistiske mål, da tiden til projektet er begrænset til fire måneder og 5 ECTS point. Da der er fokus på at arbejde iterativt i dette semesterprojekt, stod det dog klart at disse afgrænsninger kunne ændre sig i takt med projektets forløb. Dog blev vi enige om, at produktet skulle være en webapplikation, som udvikles på baggrund af at blive tilgået fra en stor monitor (typisk PC), og kun hvis der var tid tilovers, ville fokus være at skabe et design som også kunne tilgås fra mobile enheder - dette har dog ikke været aktuelt.

Kernefunktionaliteten i en quiz-applikation som denne, må vel betragtes at være besvarelsen af en given quiz. Kravspecifikationen og use casene blev udviklet på baggrund af denne tankegang, og det blev derfor vedtaget at sætte fokus på følgende features:

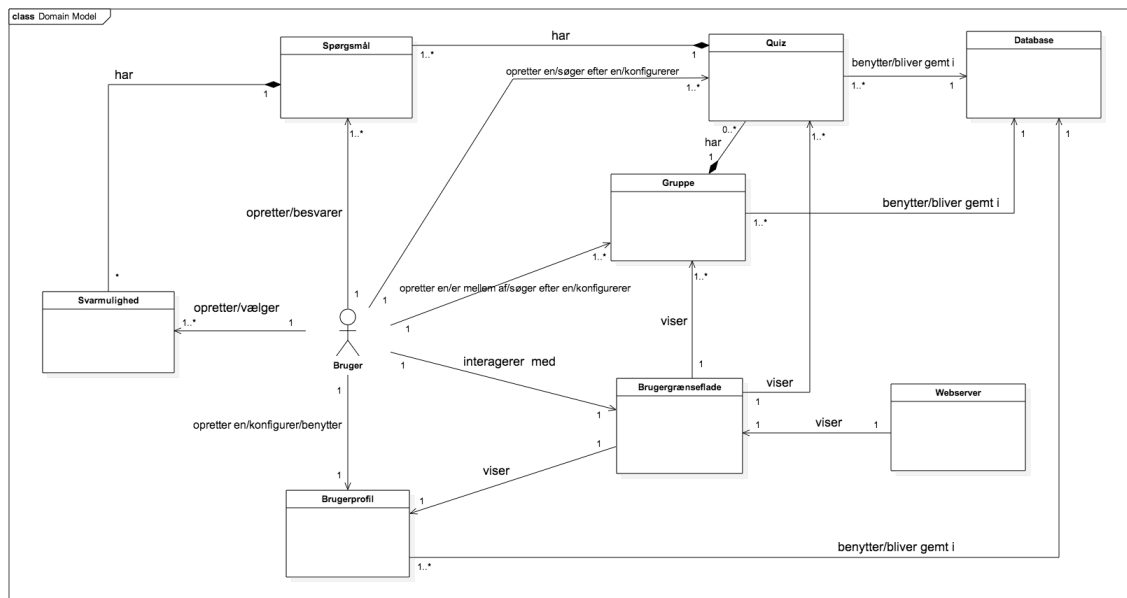
- At skabe en brugergrænseflade som er intuitiv at manøvrere igennem.
- Kunne oprette en bruger og logge ind/ud.
- Kunne oprette en quiz.
- Kunne ændre i denne quiz på et senere tidspunkt, f.eks. tilføje flere spørgsmål.
- Kunne oprette en gruppe og tilføje quizzer til denne.



# 4

## Identificering af problemdomæner

Domænemodellen på figur 4.1 er udviklet efter at den overordnede systemarkitektur er lagt på plads. Den fungerer som en visuel ordbog over projektet. De konceptuelle klasser er identificeret ud fra use casene som er diskuteret i idé-fasen. Ud fra domænemodellen kan man analysere sig nærmere ind på systemets problemdomæner.



Figur 4.1. Domænemodel over det samlede system

Ud fra ovenstående Domænemodel er der blevet identificeret følgende problemdomæner:

- Webserver
  - Webserver præsenterer brugergrænsefladen for bruger.
- Brugergrænseflade
  - Bruger interagerer med brugergrænsefladen.
- Gruppe
  - Bruger kan oprette en eller flere grupper. Hver gruppe kan indeholde en eller flere quizzes.
- Quiz
  - Bruger kan oprette en eller flere Quizzes Hver quiz har en eller flere spørgsmål.
- Brugerprofil

- Bruger kan oprette sin egen brugerprofil.
- Database
  - Databasen gemmer quizzer, grupper og brugerprofiler.

# 5

## Iteration 1

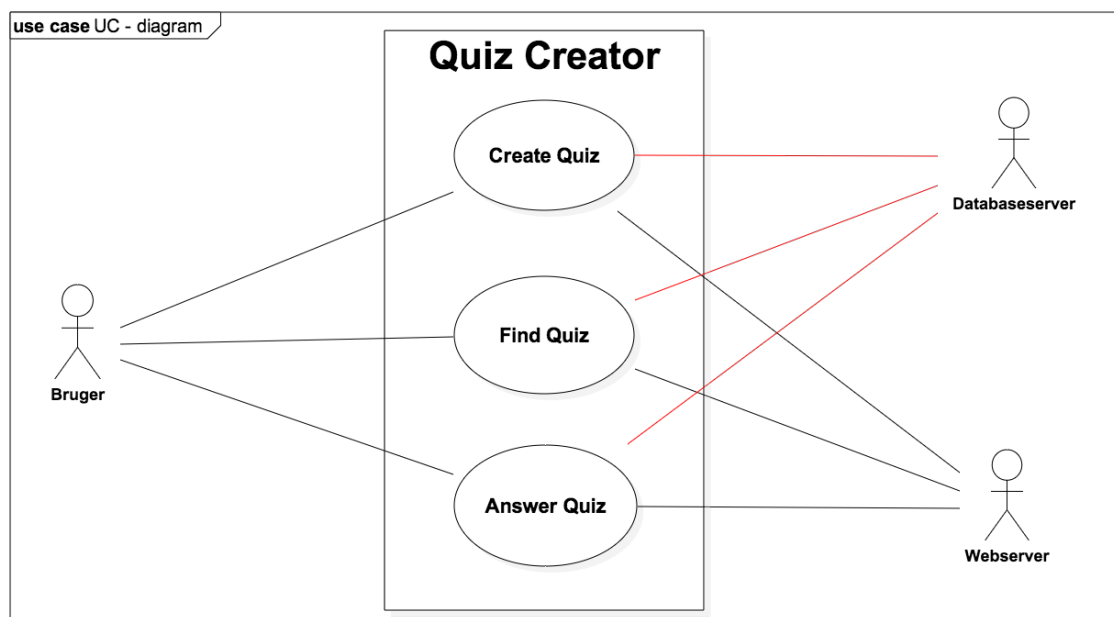
I første iteration bestod størstedelen af arbejdet i at opsætte website og database. Vi valgte derfor kun at medtage de meste fundamentale funktioner. Disse funktioner var at oprette en quiz, søge efter en quiz og gennemføre en quiz. Ud over at implementere og beskrive disse use cases var en stor del af iteration 1, at designe hvordan websiden skulle se ud, og foretage en forundersøgelse af hvilke frameworks der skulle gøres brug af. Disse emner er uddybet i de følgende afsnit.

### 5.1 Krav

Kravende til første iterations produkt er specificeret i punktform her under:

- Det skal være muligt for en bruger af systemet at oprette en quiz.
- Quizzen skal indeholde ét spørgsmål.
- Spørgsmålet skal have to svarmuligheder, hvor mindst ét er korrekt.
- Det skal være muligt at finde en quiz ved at søge efter quizzens tags eller navn.
- En bruger af systemet skal kunne svare på den oprettede quiz.

Disse krav blev omformuleret til tre use cases med tilhørende aktører, som vist på figur 5.1



*Figur 5.1.* Identificerede aktører og deres forbindelser til use cases

## Ikke-funktionelle krav

Ud over de funktionelle krav, der beskrives i use cases, blev der også defineret nogle ikke-funktionelle krav, som kan ses i følgende afsnit. Disse krav blev til dels udarbejdet for at danne en fælles forståelse for nogle af systemets rammer.

## Brugbarhed

- Programmet skal virke på Mozilla Firefox 35 og nyere (PC og Mac)
- Quiz-navnelængde er minimum 1 og maksimalt 75 karakterer
- Spørgsmål-navnelængde er minimum 1 og maksimalt 75 karakterer
- Tags-navnelængde er minimum 1 og maksimalt 75 karakterer
- Det er muligt at tilføje 0 til 10 tags til en Quiz

## Ydeevne

- Programmet skal kunne afvikles af minimum 10 samtidige brugere, som hver har 1 session åben

## 5.2 Forundersøgelse

### Grafiske brugerflade-teknologier

Følgende teknologier er vurderet i forbindelse med udviklingen af den grafiske brugerflade.

Navn	Pros	Cons
Microsoft WFP	<ul style="list-style-type: none"><li>• Anvendes i I4GUI</li></ul>	<ul style="list-style-type: none"><li>• Kan kun bruges på Windows-platfomen</li><li>• Kræver .NET-frameworket på klientens computer</li></ul>
Microsoft Silverlight	<ul style="list-style-type: none"><li>• Stærkt framework til grafisk præsentation</li></ul>	<ul style="list-style-type: none"><li>• Udvikles ikke længere af Microsoft</li><li>• Kræver Silverlight-frameworket på klientens computer</li></ul>
HTML5 og CSS3	<ul style="list-style-type: none"><li>• Anvendes i I4GUI</li><li>• Den gængse teknologi for web-applikationer</li><li>• Understøttet af de fleste platforme</li></ul>	<ul style="list-style-type: none"><li>• Mindre grafiske evner</li></ul>

Ud fra ovenstående er HTML5 og CSS3 valgt for at få en bredt undersøgt teknologi som har rigelige muligheder for programmets behov og anvendes generelt mest af de belyste teknologier.

## Database-teknologier

Følgende teknologier er vurderet i forbindelse med persistering af data.

Navn	Pros	Cons
Microsoft SQL Server	<ul style="list-style-type: none"><li>• Anvendes i I4DAB</li><li>• Gratis server igennem Microsofts Azure [Microsoft, 2015b]</li><li>• Integreret med Entity Framework</li></ul>	<ul style="list-style-type: none"><li>• Licenseret brug</li></ul>
MySQL	<ul style="list-style-type: none"><li>• Meget anvendt inden for webudvikling i sammenhæng med PHP</li><li>• Webbaseret GUI med phpMyAdmin [Müller et al., 2015]</li><li>• Open-source</li></ul>	<ul style="list-style-type: none"><li>• Kompatibilitet med ASP.NET MVC er begrænset</li><li>• Integration med Entity Framework begrænset</li></ul>
Oracle Database	<ul style="list-style-type: none"><li>• Integration med Entity Framework</li></ul>	<ul style="list-style-type: none"><li>• Licenseret brug</li></ul>

Ud fra ovenstående er Microsoft SQL Server valgt. Dette er begrundet af muligheden for at få adgang til en server igennem Azure samt at ASE stiller en server til rådighed. Da undervisningen i faget I4DAB også anvender denne teknologi er det et oplagt valg.

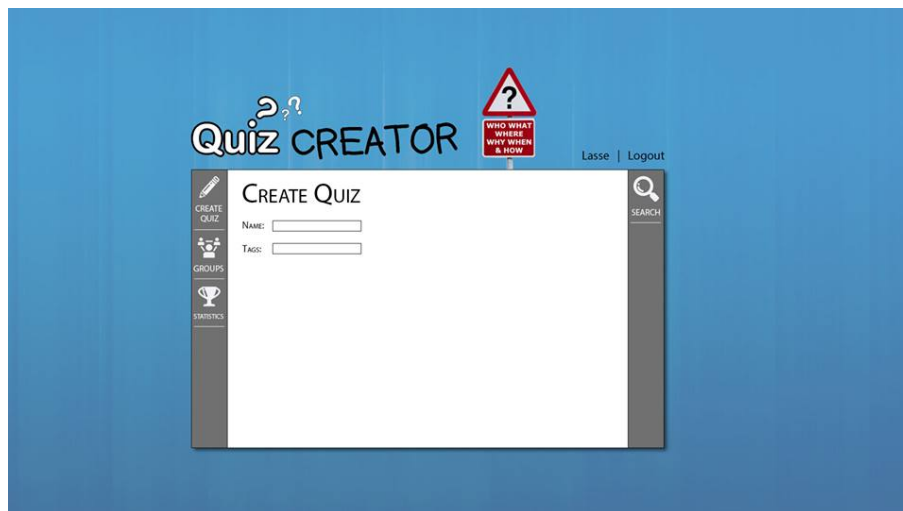
## 5.3 Grafisk brugergrænseflade design

Brugeren interagerer med applikationen gennem brugergrænsefladen, og det er derfor vigtigt at denne indeholder simple og responsive menuer, der designmæssigt også gerne må være flotte og appelerende. Gennem første iteration er en del GUI blevet præsenteret og diskuteret i mellem projektgruppens medlemmer. Første udkast ses på figur 5.2.



**Figur 5.2.** Skitse 1 for systemets GUI

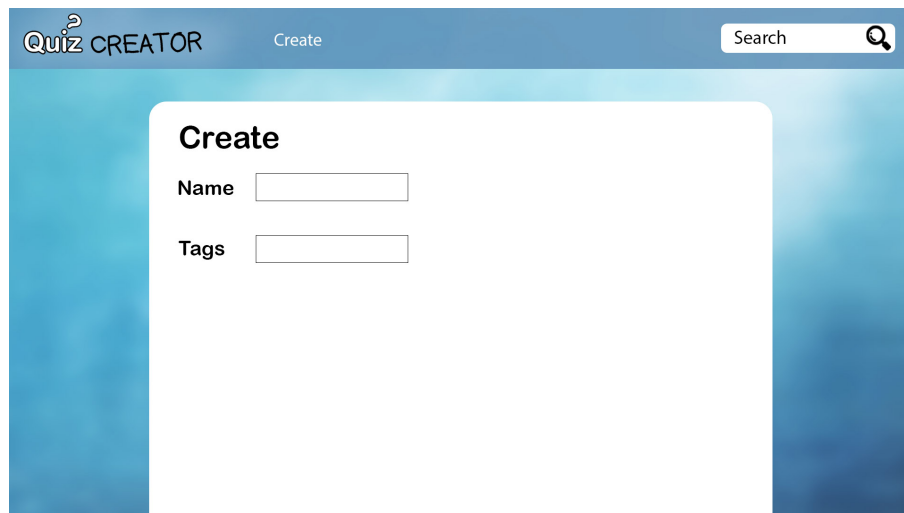
Dette er en løs skitse, som blot blev brugt til at give gruppen en idé om i hvilken retning projektet skulle skride frem. Det blev dog vedtaget at farvetemaet skulle være lysere. På baggrund af disse overvejelser blev skitse nummer to, set på figur 5.3 udarbejdet.



**Figur 5.3.** Skitse 2 for systemets GUI

Efter gennemgående overvejelser blev det besluttet at "åbne" designet og benytte hele skærmen og udnytte skærmens top til en bjælke.





**Figur 5.4.** Skitse 3 for systemets GUI. Dette er den endelige skitse.

Iterationens endelige designskitse ses på figur 5.4. En bjælke løber langs toppen, og giver brugeren en "quick-access" menu til til at oprette og søge efter eksisterende quizzes. Figuren viser kun et udkast af designet eftersom der skal tilføjes flere muligheder i indtastningerne mv.

## 5.4 Arkitektur

Da applikationen skal være en webapplikation, har der været nogle overvejelser i forhold til hvilken arkitektur der skal anvendes. Microsoft tilbyder et framework kaldet ASP.NET MVC [Microsoft, 2014]. Dette har vi anvendt da det tilbyder en stærk funktionalitet og er bredt anvendt inden for webudvikling. Det bygger på Model-View-Controller-arkitekturen der deler funktionaliteten op i tre logiske lag. View-laget holder HTML-siderne og tilhørende CSS formatering, altså fremvisningen af data. Den har også simpel viewafhængig logik, som f.eks. at skjule knapper hvis brugeren ikke er logget ind. Model-laget integrerer med Entity Framework [Microsoft, 2015a] samt holder View-Modeller som er viewafhængige modeller. Controllerne styrer funktionaliteten for hver use case og indeholder al business logikken.

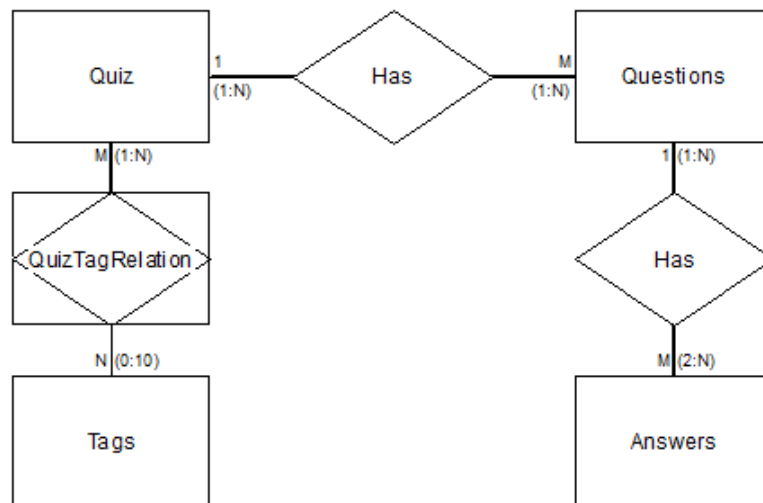
## 5.5 Database

I denne iteration var der fokus på at få gemt brugerens oprettede quizzes, samt at kunne finde dem igen via quizzens tags, som search-funktionen vil søge på.

ASE stillede en MS SQL server [Microsoft, 2015d] til rådighed, som blev brugt som løsning til database problematikken. Denne databaseserver er placeret på et internt netværk på ASE hvilket kræver at man er logget på ASE's VPN for at få adgang til den. I programmet DDS-Lite, som blev introduceret i 4. semesterets databasekursus I4DAB, udvikledes et ER-diagram, vist på figur 5.5, som blev benyttet til at opsætte et databaseschema.

Schemaet indholder entiteterne Quiz, Question, Answer, Tag og QuizTagRelation. QuizTagRelation entiteten, er en mange-til-mange relation som indeholder hhv. et Quiz-

og TagId



**Figur 5.5.** ER-diagram over databasen i 1. iteration

For at overholde princippet om separation of concerns, blev der lavet et DAL og en QuizModel. DAL blev implementeret med ADO.NET-frameworket, som muliggjorde at queries blev sendt til databasen. Dette DAL åbnede forbindelsen til databasen når der var brug for det, og lukkede den igen efter operationen var blevet udført. ADO.NET blev valgt på baggrund af en introduktion i kurset I4DAB.

QuizModel-klassen er en facade-klasse i mellem DAL og controllerene. Her lægges logik som håndterer komplekse database-forespørgsler, som når en komplet quiz skal hentes med tilhørende spørgsmål og svar. Det samme gælder, når quizzes skal gemmes mv.

## 5.6 Dokumentationsværktøjet Doxygen

For at lette dokumentationsarbejdet i forbindelse med koden til applikationen, blev det valgt at anvende dokumentationsværktøjet Doxygen [Heesch, 2015]. Programmet kører kildekoden igennem og registrerer specielt-noterede kommentarer i koden. På baggrund af disse genereres en hjemmeside som kan bruges til at danne overblik over opbygningen og relationerne i mellem klasserne i systemet.

Doxygen fungerer på flere platforme og forskellige programmeringssprog. I dette projekt er C# anvendt. Med Doxygen skrives dokumentationen i selve koden, dette muliggør at skrive sin dokumentation samtidig med at man udvikler sin kode. Ydermere bliver dokumentationen overskuelig i den forstand at dokumentationen for den pågældende klasse eller metode står sammen med den funktionelle kode og er derfor let at vedligeholde.

## 5.7 Sprints

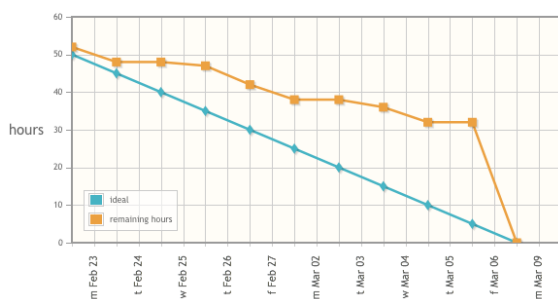
I denne iteration var der tre sprints. Det første sprint var én uge og de to andre to uger. Nøgletallene for iterationen er anført i tabel 5.1

Sprint	Opgaver	Forventede timer	Registrerede timer	Dækning
1	16	17	9	52 %
2	22	51	32	62 %
3	34	83	61	73 %

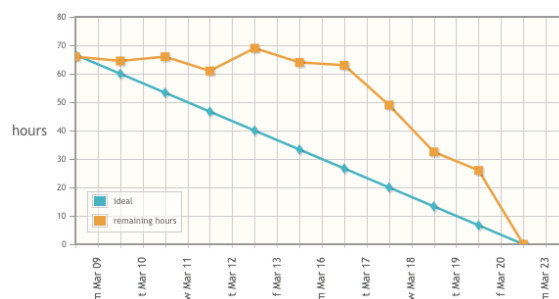
**Tabel 5.1.** Tabel med nøgletal for hvert sprint i iteration 1

Mængden af arbejde der blev taget ind steg væsentligt over de tre sprints. Vi skulle lige i gang med Scrum-tankegangen, finde ud af hvor høj en belastning vi kunne arbejde med og have en god måde at både estimere og bruge timerne på.

Logningen af timer fungerede ikke optimalt. Ud fra tallene brugte vi kortere tid på at nå i mål med sprintene, men i virkeligheden fik vi ikke logget det rigtige antal timer. På graferne på figur 5.6 og 5.7 vises vores time-burndown i løbet af de to sidste sprints. De viser at vores timer hovedsagligt er brugt i slutningen af hvert sprint. Det pludselige dyk til sidst er udtryk for at opgaverne ikke er blevet logget, men alligevel er løst til tiden.



**Figur 5.6.** Burndown-graf for sprint 2

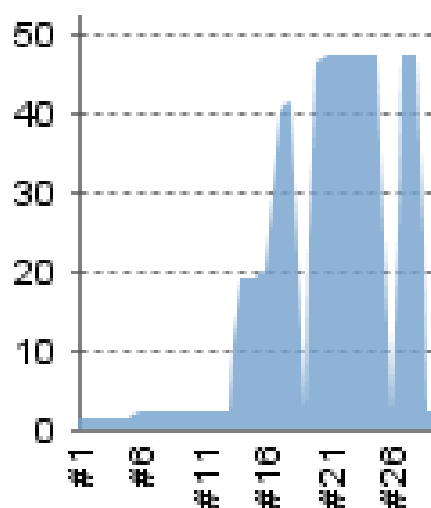


**Figur 5.7.** Burndown-graf for sprint 3

## 5.8 Test

På baggrund af de indledende arkitektske beslutninger omkring at anvende ASP.NET MVC frameworket, åbner muligheden for at anvende NUnit [Poole et al., 2014] til at teste controllerne i systemet.

For at udnytte disse tests til fulde kombineres de med et Continuous Integration (CI)-system. ASE stiller en Jenkins-server [Kawaguchi, 2015] til rådighed for dette, hvorfor det var et oplagt valg. Systemet er sat op til at køre alle tests ved hvert commit i kode-basen på Git-serveren. På figur 5.8 er et udsnit af kørte tests på Jenkins-serveren for første iteration. De første sprints genererede ikke meget kode, hvorfor der ikke er mange tests der, men efter build 13 begynder der at komme en del på.



*Figur 5.8.* Automatiskudførte tests på Jenkins-server for iteration 1

I anden iteration skulle der tilføjes noget ekstra funktionalitet på websiden, og vi valgte at tilføje mulighed for at oprette en bruger og logge ind siden. Herved skabes et ejerskab over quizzet, og der gives mulighed for at konfigurere sine quizzet, hvis det skulle ønskes. Derudover valgte vi at refakturere vores DAL til Entity Framework.

## 6.1 Krav

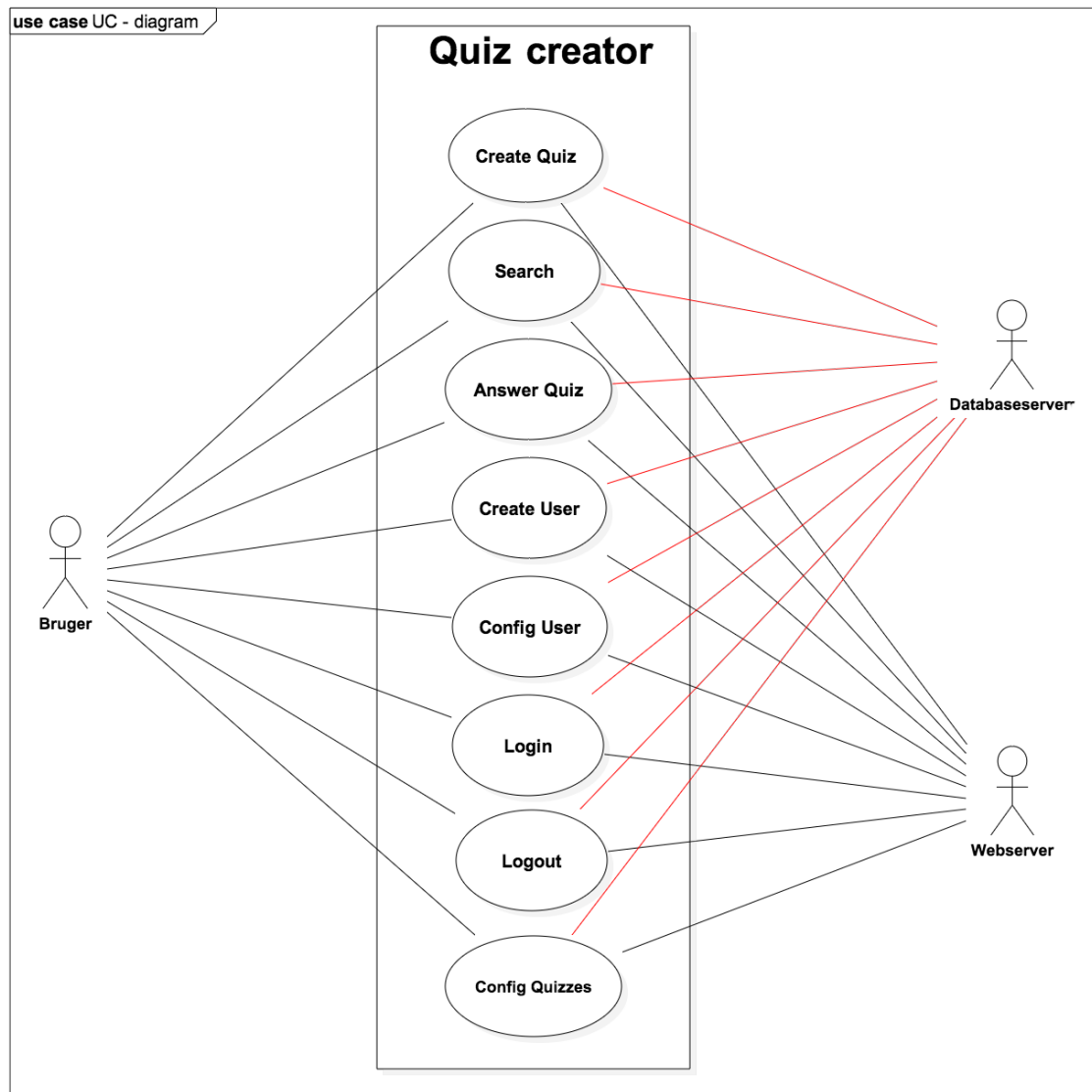
I anden iteration i projektet skulle produktet udvides, og den mest logiske udvidelse ville være at tilføje brugere til systemet. I første iteration var intet personligt: Hvis man åbner websiden og opretter en quiz tilhører den ikke nogen, men alle kan se og besvare den, samtidig kunne man ikke ændre quizen, hvis den blev oprettet med en fejl eller skulle udvides. I anden iteration blev der derfor lavet en liste over krav, som gav anledning til nye use cases:

- Det skal være muligt at registrere sig som bruger i systemet.
- Man skal kunne logge ind som bruger eller med sin Facebook konto.
- Bruger-data som navn, password mm. skal kunne ændres af brugeren.
- Brugernavn og email skal være unikke.
- Efter login skal man kunne administrere de quizzet, som man har oprettet.

Derudover blev der opsat et par punkter, som medvirker ændringer i tidligere use cases:

- Man skal kunne uploade et billede til hvert spørgsmål i en quiz.
- Mulighed for at tilføje flere spørgsmål til en quiz.
- Mulighed for at tilføje flere svarmuligheder til et spørgsmål.

Disse krav medførte et opdateret use case diagram, som ses på figur 6.1.



*Figur 6.1.* Use case diagram for iteration 2

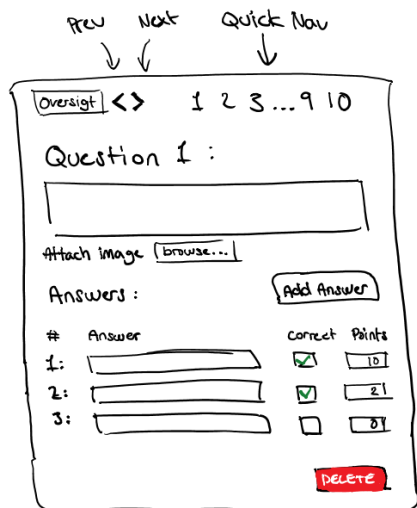
## 6.2 Ikke-funktionelle krav

Til denne iteration blev de ikke-funktionelle krav opdateret, så de passer bedre til virkelighedens brugsscenarier: Gældende for både quiznavn, spørgsmål, svarmuligheder og tags er nu begrænset til minimum 1 karakter for svarmuligheder og 2 for quiznavn og spørgsmål. Dette skyldes at test-input for disse har vist at det passer bedre til virkelighedens behov (f.eks. at kunne angive '1' som svar til spørgsmålet "hvad er 4-3?"). Der er også oprettet ikke-funktionelle krav til tekstlængden på brugernavn og adgangskode, hhv. fra 2 til 75 karakterer og 6 til 75 karakterer. Derudover skal både brugernavne og emailadresser skal være unikke.

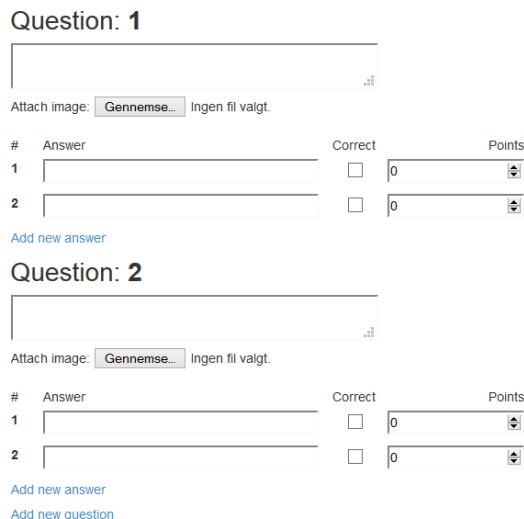
## 6.3 Config og Create Quiz

Før vi gik i gang med at implementere use casene "Config Quiz" og "Create Quiz", havde vi gjort det klart hvordan den grafiske brugergrænseflade skulle se ud for de to. Opbygning

skulle nemlig være identisk, da de begge behandler en quiz. På figur 6.2 ses en skitse for hvordan vi ville have GUI til at se ud, og på figur 6.3 ses den grafiske brugerflade for Create Quiz. Det ses at i den nuværende iteration ikke er blevet tilføjet mulighed for at bladre gennem spørgsmålene, men blot får dem listet ud efter hinanden.



**Figur 6.2.** Create og Config Quiz skitse



**Figur 6.3.** Create Quiz grafisk opbygning

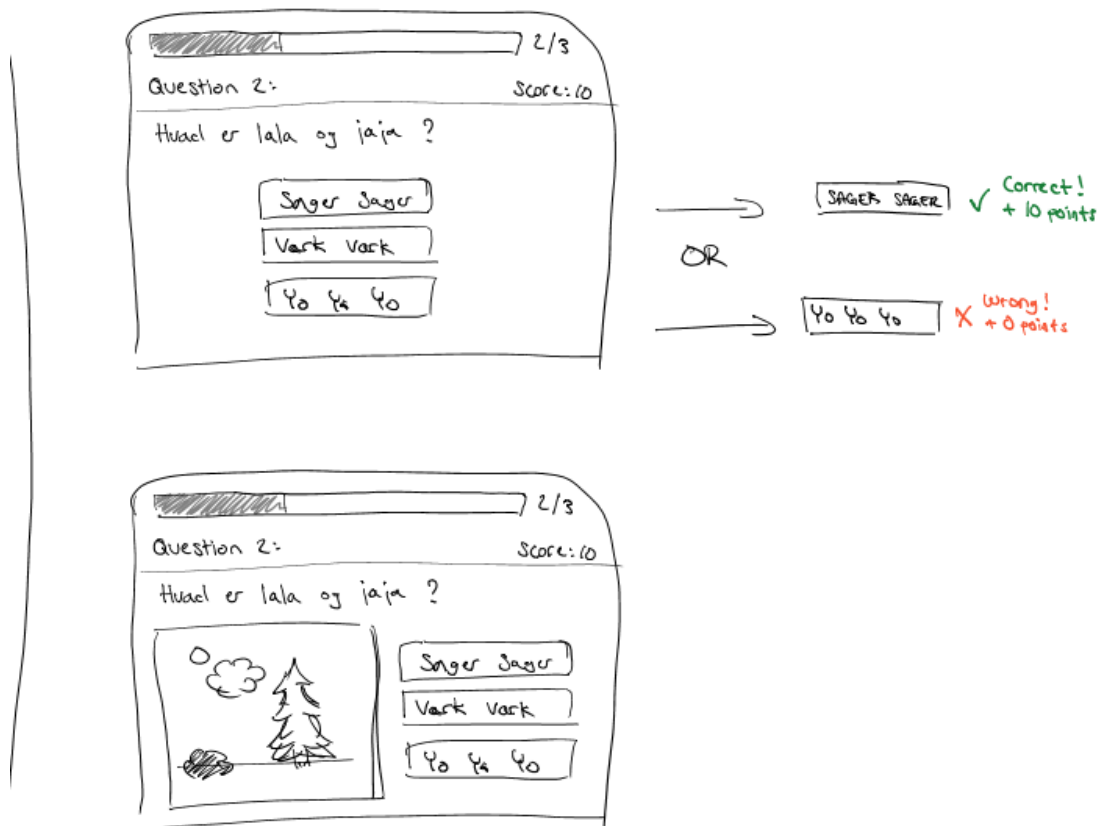
Vi havde brug for at udvide funktionaliteten sådan at en bruger dynamisk kunne tilføje flere spørgsmål og svarmuligheder.

For at gøre dette dynamisk anvendes en AJAX løsning, hvor nye inputfelter kan oprettes i browseren uden siden skal genindlæses. Selve AJAX delen er et simpelt javascript, som tilgår en metode i en controller som returnerer den nødvendige HTML-kode til at redigere et spørgsmål eller svar. Denne returnerede HTML-kode indsættes efter de eksisterende svar eller spørgsmål. ASP.NET tilbyder en funktionalitet kaldet model binding, hvilket betyder, at navnene på inputfelter indikerer deres relation til et objekt i controlleren. Dette gør det nemt at tilføje spørgsmål mv. og stadig kun have et argument til controlleren.

## 6.4 Answer Quiz

At kunne besvare en quiz er et essentielt element og "flaskehalsen" for projektet. Hvis brugergrænsefladen til besvarelsen af quizzes ikke føles let og overskuelig vil QuizCreator hurtigt tabe til eventuelle konkurrenter. På figur 6.4 ses den indledende designskitse af hvordan brugergrænsefladen til use casen Answer Quiz ser ud.

## Answer Quiz



Figur 6.4. 1. skitse af Answer Quiz brugergrænsefladen.

Øverst i quiz-vinduet ses progressbaren, som giver hurtigt overblik over, hvor langt brugeren er nået i quizzzen. I centrum ses det respektive spørgsmål som er i gang med at blive besvaret. Af denne skitse fremgår det at svarmulighederne er "knapper" der trykkes på, men efter en række designdiskussioner stod det klart, at svarmulighederne i stedet skulle implementeres som radio-buttons, altså boxe som kan krydses af. Dette blev vedtaget da det så nemt ville være muligt at se, hvilket svar man havde valgt, hvis man ville gå tilbage til et foregående spørgsmål. Answer Quiz er ikke beskrevet i 1. iteration da det var en meget simpel funktionalitet den skulle indeholde. Her skulle en quiz blot bestå af et spørgsmål som skulle kunne besvares. I denne iteration er der blevet fyldt lidt mere funktionalitet på, og bl.a. er der benyttet AJAX til at sørge for at blot spørgsmåls-delen af siden reloades når man går til et nyt spørgsmål.

## 6.5 Database

### Fokuspunkter:

- Refaktorere Data Access Layeret (DAL)
- Tilføje nye entiteter til den fysiske database



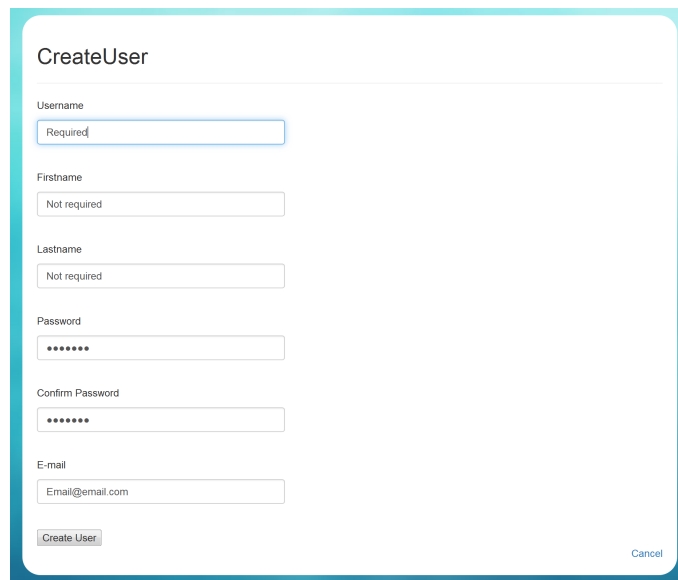
Under denne iteration blev der gjort kendskab til Entity Framework (EF) igennem semesterets kursus om databaser. Entity Framework er et Framework for ADO.NET og en af de mange Frameworks fra .NET. Entity Framework kan beskrives som et lag, der ligger oven på ADO.NET, som har til formål at skjule de fysiske database detaljer for programmøren. Hvis man sammenligner Entity Framework med ADO.NET, så har Entity Framework en lav kobling mellem koden og den fysiske database, da man ikke kan genfinde f.eks. kolonnenavne eller deres indbyrdes rækkefølge i koden. I ADO.NET er der derimod en høj kobling, da der anvendes SQL, som jo refererer direkte til den fysiske database. Fordelen ved at anvende Entity Framework er, at man som programmør arbejder på et højere abstraktionsniveau, hvor man ikke har behov for at kende strukturen for den fysiske database.

Ved brug af EF kan man skrive database applikationer uden at anvende SQL-sætninger, men i stedet skal man anvende LINQ, som er noget nemmere at bruge. Med de fordele EF tilbyder, blev der taget det valg at refaktorere det daværende DAL fra at anvende ADO.NET til EF, og Quizmodel klassen omskrives til at bruge det nye DAL med EF.

Udover at refaktorere DALet er der tilføjet nye entiteter til den fysiske database som skal gemme brugerdata. Disse entiteter er genereret af ASP.NET identity 2.0 Framework, og beskrives nærmere i afsnit 6.7.

## 6.6 Users

Under denne iteration blev der tilføjet endnu to vigtige funktionaliteter - nemlig oprettelse af en bruger samt konfigurere en bruger. Til at håndtere brugerprofiler, har vi anvendt ASP.NET Identity framework 2.0 [Microsoft, 2015c]. Dette er et bibliotek som har mange brugbare klasser, når det gælder håndtering af brugerprofiler. Klasser som gør det muligt at oprette en ny bruger med en bestemt adgangskode, slette en bruger, opdatere en bruger, ændre adgangskoden til en bruger, sende en E-mail og/eller SMS til en bruger og meget mere. Et andet argument for at anvende frameworket, er at det selv sørger for at hashe brugerens adgangskode. På denne måde sikres dataen i databasen, så uvedkommende ikke kan få adgang til de oprindelige koder. På figur 6.5 ses brugergrænsefladen for oprettelse af en bruger.



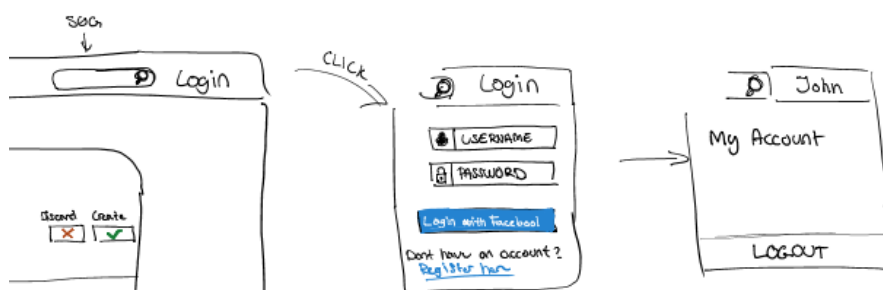
The screenshot shows a web form titled "CreateUser". It contains several input fields: "Username" (marked "Required"), "Firstname" (marked "Not required"), "Lastname" (marked "Not required"), "Password" (masked with dots), "Confirm Password" (masked with dots), and "E-mail" (with a placeholder "Email@email.com"). At the bottom left is a "Create User" button, and at the bottom right is a "Cancel" link.

**Figur 6.5.** Brugergrænseflade for Create User viewet

Når der oprettes en ny bruger, er der krav om at brugeren skal vælge en brugernavn og en adgangskode samt oplyse en gyldigt e-mail-adresse. Brugeren kan også udfylde sit navn, men det er ikke et krav. Brugeren kan til hver en tid omkonfigurere sin profils informationer. Dog har vi valgt ikke at give brugeren lov til at ændre brugernavnet når denne er oprettet. Det er bestemt at brugernavne skal være unikke og til at sørge for at en bruger ikke kan vælge et brugernavn som allerede er i brug, er der brugt remote-validation, se dokumentationen afsnit 7.3.

## 6.7 Login system

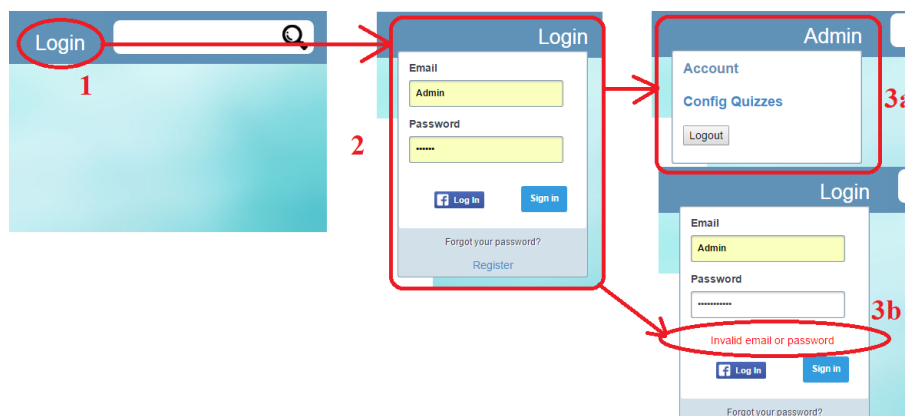
I denne iteration kom login-systemet på banen. Der blev valgt at man skulle kunne logge ind med en identitet med tilhørende adgangskode, eller via Facebook. Det første udkast til brugergrænsefladen for login ses på figur 6.6. Her ses det at man benytter et brugernavn, men grundet Facebook, er dette blevet ændret til email.



**Figur 6.6.** Login GUI skitse

Login knappen aktiverer en dropdown menu, hvor i man indtaster email og adgangskode. Med Facebook login knappen kan der logges ind uden at skulle indtaste nogle oplysninger, hvis man altså er logget ind på Facebook. Hvis man ikke er logget ind på Facebook vil

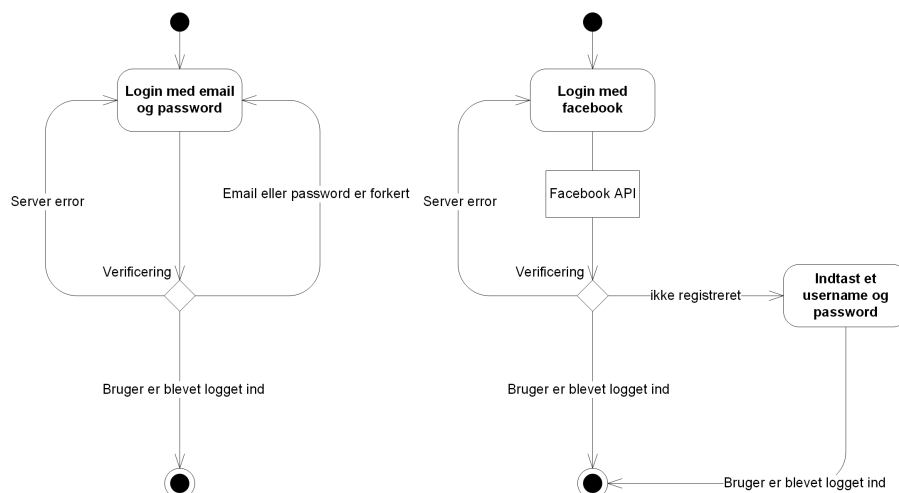
facebook's API bede om at man logger ind på Facebook. Scriptet til dropdown menuen er skrevet i JavaScript. Login-funktionerne bruger AJAX med tilhørende API controllers til at logge brugeren ind.



**Figur 6.7.** Den endelige Login GUI

Figur 6.7 illustrerer hvordan login blev implementeret på siden. Her udskiftes login knappen i menuen med brugerens brugernavn, hvor brugeren nu har muligheden for at konfigurere sine brugeroplysninger.

Login systemet bruger .NET's Identity 2.0 frameworket til autentificering. Frameworket genererer ligeledes nogle default database-entities, som vi valgte at bruge i denne iteration for at spare tid. Disse kan ses i dokumentationens afsnit <sup>1</sup>, hvor de alle indeholder "AspNet" i sit navn. Når brugeren er autoriseret, bliver der gemt en cookie hos klienten så denne genkendes.



**Figur 6.8.** Aktivitetsdiagram over de to login metoder: Login med email og login med Facebook

Figur 6.8 viser hvordan loginsekvensen foregår bag facaden. Første gang en bruger benytter "Login med Facebook" vil personen blive bedt om at indtaste brugernavn og adgangskode. Brugernavnet vil kun fungere som brugerens navn/identitet udadtil på siden.

<sup>1</sup>FiXme Note: Lars: Indsætte reference til dokumentationens dataview afsnit

Adgangskoden vil kunne bruges til at logge ind på siden uden at benytte Facebook login. Der er implementeret fejlhåndtering af eventuelle serverfejl der kunne opstå. Hvis der opstår en serverfejl, vil brugeren få en fejlmeddelelse.

**Udfordringer:** Der opstod nogle udfordringer, da vi arbejdede på den interne webserver, som er stillet til rådighed af Ingeniørhøjskolen (IHA) ved Aarhus Universitet.

- Det første problem opstår, når man forsøger at registrere den interne webservers URL på Facebook. Da webserveren har en intern IP-adresse på IHAs netværk, kan Facebooks servere ikke få adgang til denne. Dette problem står til at blive løst i en anden iteration. Muligvis med en Azure hosted webserver.
- Det andet problem opstår ligeledes pga. den interne server. 10.29.10.30/QuizCreator er den URL på vores hjemmeside tilgås fra, når den er published. AJAX, som kalder login controlleren, tror at root er 10.29.10.30 istedet for at være 10.29.10.30/QuizCreator. Dette kan godt fixes, dog vil det kun virke på enten localhost eller den interne server. Her er løsningen igen en anden webserver.

## 6.8 Sprints

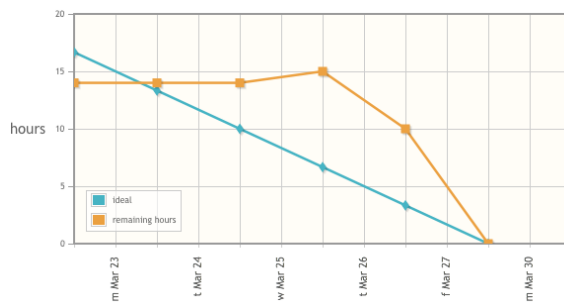
I denne iteration var der tre sprints. Det første og det sidste var på én uge og sprint nummer to var på to uger. Nøgletallene for iterationen er anført i tabel 6.1

De forventede timer i hvert sprint er højere end sidst. Dette skyldes at vi gerne ville nå længere i denne iteration.

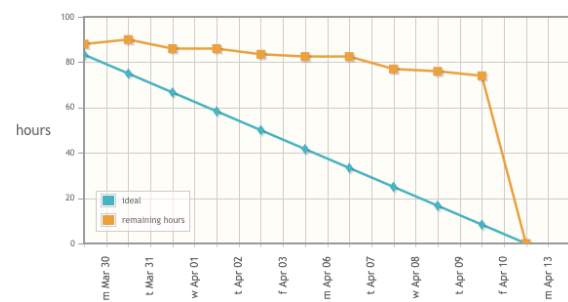
Vi forsøgte i denne iteration at være mere fokuseret på at få registreret de brugte timer, så vi kunne se om vores estimeringer var retvisende. Dette lykkes til dels og dog nåede vi ikke helt i mål. En hel use case blev flyttet til næste iteration, og derfor er time-dækningen ikke helt retvisende. Vi havde forventet at hver opgave tog kortere tid end de endte med at tage, og var derfor nødt til at prioritere denne use case lavere. Dette kan også ses på burndown-graferne på figur 6.9, 6.10 og 6.11. Generelt når vi ikke alle de timer som det var forventet og arbejdet bliver skubbet til de sidste par dage. Specielt for sprint 3 kan man også se hvordan antallet af timer stiger i starten af sprintet. Dette skyldes at vi ikke havde estimeret alle opgavernes længde inden vi begyndte sprintet. Af samme grund er grafen for sprint 2 også meget vandret fordi vi har estimeret opgaver, løst dem og lukket dem i mens sprintet er i gang.

Sprint	Opgaver	Forventede timer	Registrerede timer	Dækning
1	15	23	15	65 %
2	48	80	77	96 %
3	29	45	28	62 %

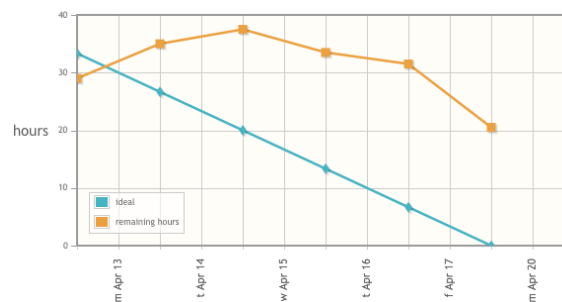
**Tabel 6.1.** Tabel med nøgletal fra iteration 2



Figur 6.9. Burndown-graf for sprint 1



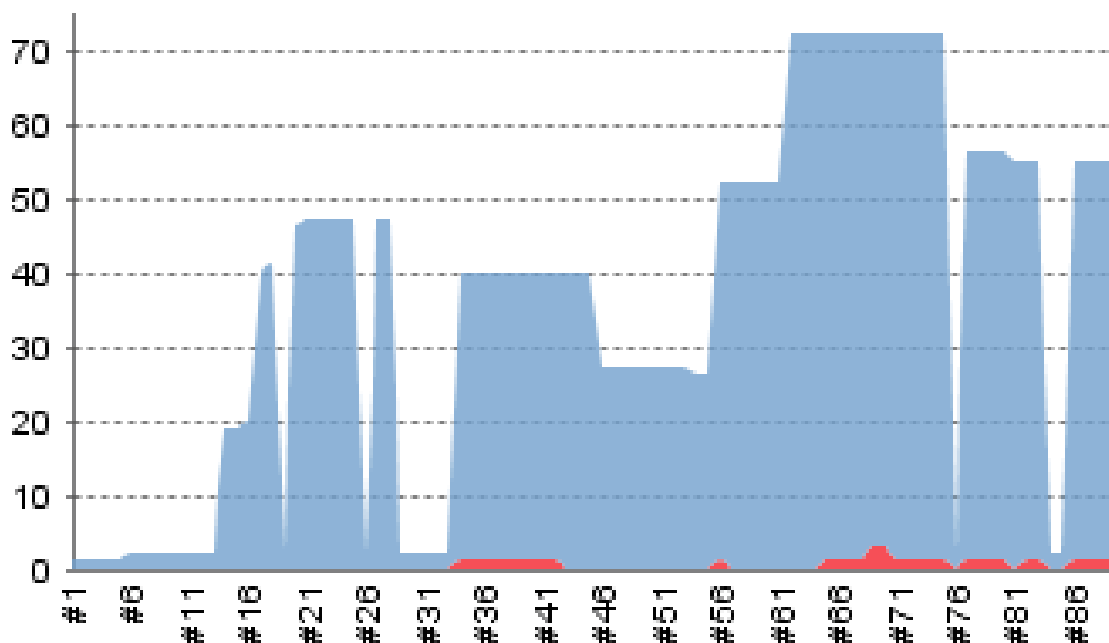
Figur 6.10. Burndown-graf for sprint 2



Figur 6.11. Burndown-graf for sprint 3

## 6.9 Test

Efter iteration 2 er der kommet godt gang i Jenkins-serveren. Alle controllere har tilknyttet en række unit-tests for løbende at sikre at færdige funktioner ikke går i stykker når der tilføjes nye funktioner. På figur 6.12 er et udklip af de udførte tests for starten af projektet og frem til slutningen af iteration 2.



Figur 6.12. Automatiskudførte tests på Jenkins-server for iteration 1 og 2

De skarpe dyk ved f.eks. build #76 skyldes commits af kode som ikke kan bygges. Det kan også ses at en større ændring af eksisterende kode (bl.a. ombygning i DAL) har resulteret i at omkring 20 tests ikke længere køres efter build #76.

I tredje og sidste iteration ville vi have alt den funktionalitet med som vi følte var nødvendig for at have en god og velfungerende webapplikation. Fokus har derfor været at implementere og opdatere funktioner samt at få disse til at fremstå på en præsentabel måde, som vil gøre brugeroplevelsen så god som muligt til det endelige produkt.

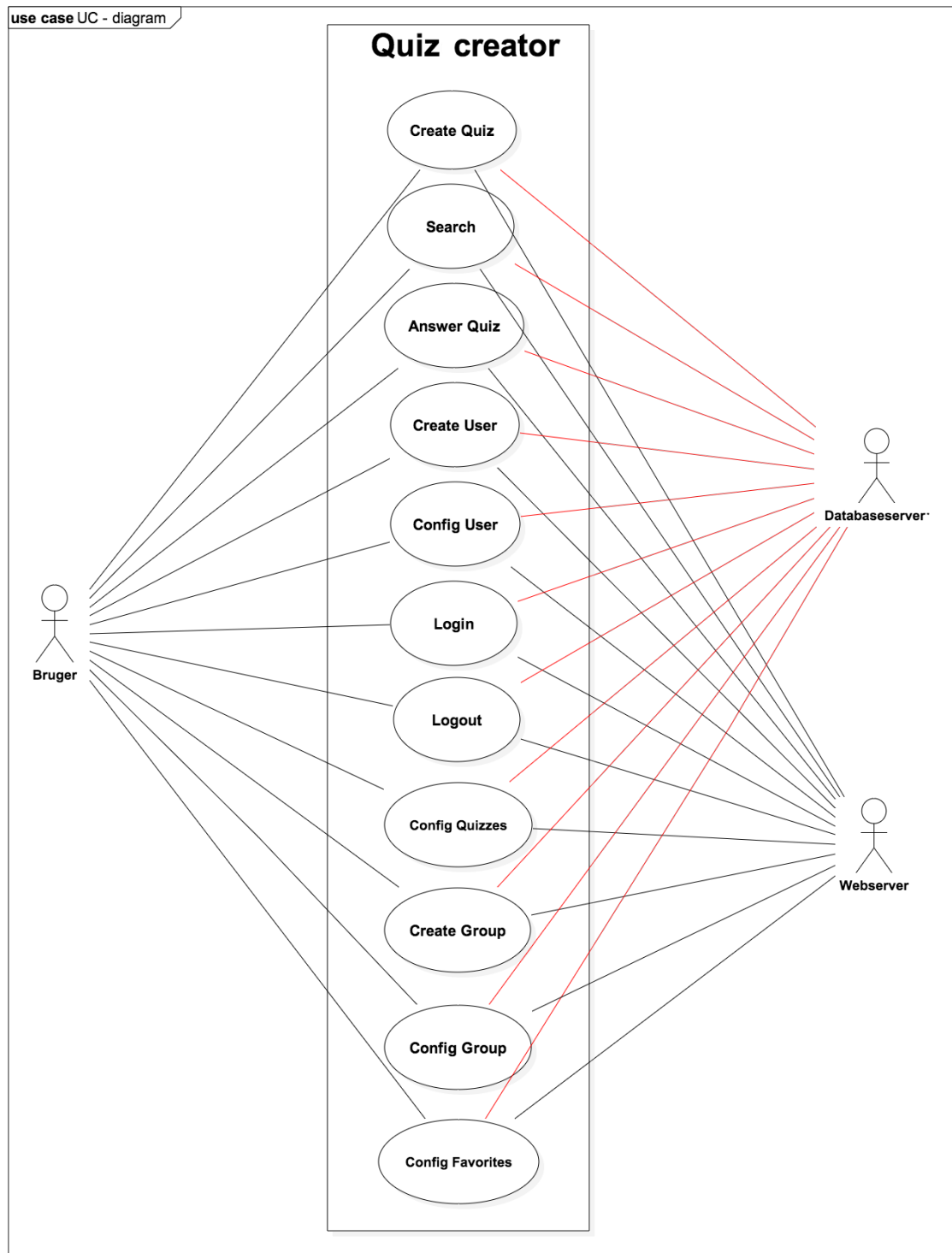
## 7.1 Krav

I iteration 3 blev systemet udvidet med en række nye features - derudover blev der videreudviklet på gamle features. Nye features bringer grupper og favoritter i spil, som hhv. er en måde at organisere quizzes og gemme quizzes og grupper til hurtig adgang. I starten af iterationen blev der opstillet en række krav til de nye funktioner:

- Mulighed for at slette spørgsmål og svar i en eksisterende quiz.
- Mulighed for at kunne oprette grupper og derefter kunne ændre i egne oprettede grupper.
- Søgning skal vise både grupper og quizzes med de søgte tags.
- Mulighed for at tilføje quizzes og grupper til favoritter ved søgning.
- Mulighed for at fjerne favoritgrupper og -quizzes.
- Mulighed for at tilføje quizzes fra søgeresultater til grupper som brugeren har oprettet.
- Mulighed for at tilføje, ændre og slette billeder på spørgsmål i en eksisterende quiz.
- Mulighed for at tilføje egne quizzes til gruppe, når gruppen oprettes.

Disse krav resulterede i tre nye use cases, samt ændring i en use case. Disse use cases ses i use case diagrammet på figur 7.1, som er det endelige use case diagram for systemet.

- UC2 Find Quiz ændres til UC2 Search (som både kan finde quizzes og grupper)
- Ny use case: Create group
- Ny use case: Config group
- Ny use case: Config Favorites



*Figur 7.1.* Use case diagram for iteration 3 og det endelige systems use case diagram

## 7.2 Answer Quiz

I denne iteration har Answer Quiz use casen igen gennemgået nogle tilføjelser. Der bliver nu vist de billeder som brugeren eventuelt har tilføjet til spørgsmålene under Create Quiz. Dette kan allerede ses på GUI skitsen fra iteration 2 på figur 6.4. Dette blev dog flyttet til denne iteration pga. opgavens størrelse ikke vil kunne nås i iteration 2. Derudover er der



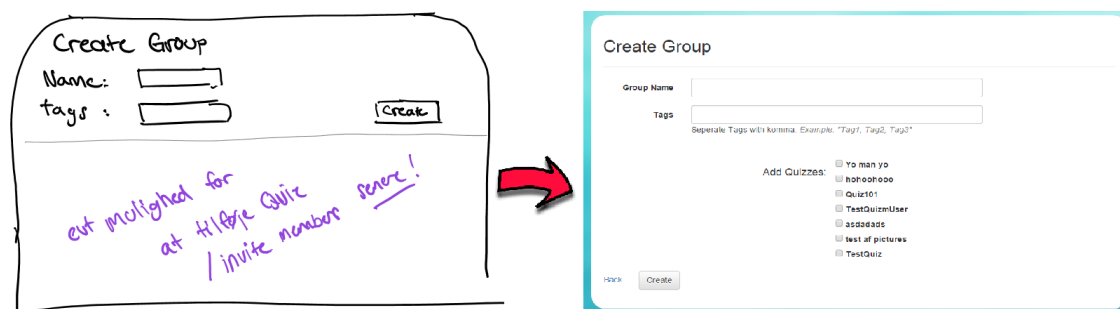
blevet tilføjet "paging" sådan at brugeren kan bladre igennem spørgsmålene og markere sine svar på radio-buttons der er stilet med Bootstrap. Som slut vises en resultat side, hvor point tælles sammen og de rigtige og forkerte svar vises.

## 7.3 Groups

En bruger kan oprette en gruppe og ligeledes blive medlem af andre grupper som er oprettet af andre brugere.

I grupper kan brugere tilføje quizzer som har relevans indenfor et bestemt emne. Det kunne f.eks. være at en bruger oprettede en gruppe der hed "2. verdenskrig" og tilføjede quizzer, der havde relevans til dette emne. Under oprettelse kan brugeren tilføje sine egne quizzer. Når man søger efter quizzer og har fundet en som har relation til sin gruppe, så kan man tilføje den til gruppen ved hjælp af en boks i søgeresultaterne.

Meningen med grupper er at brugere kan samles og dele quizzer som de mener gruppens brugere kunne have interesse i. Som nævnt kunne en gruppe indeholde quizzer specifikt til et emne, men man kunne også forstille sig en skolelærer der opretter en gruppe til klassen "2.a" og tilføjer quizzer til matematik, dansk, engelsk osv. Overordnet set er grupper altså blot et værktøj til enten at samle quizzer der har relevans for et bestemt emne eller samle brugere som quizzerne har relevans for.



Figur 7.2. Create Group fra skitse til implementering

På figur 7.2 ses til venstre det udkast der blev lavet i starten af sprintet og til højre hvordan implementeringen af dette billede er blevet.

Der er også mulighed for at konfigurere sin gruppe når den er oprettet. Her er brugergrænsefladen næsten identisk med billedet for at oprette, dog kan man ikke ændre gruppens navn, da det er besluttet at denne er bindende ved oprettelsen.

En af udfordringerne ved grupper har været at få tjekket om det valgte gruppenavn allerede var taget eller ej. Dette blev løst med model-binding og Remote Validation. Remote Validation blev ligeledes tilføjet til brugernavne så der heller ikke kunne oprettes brugere med ens brugernavne.

Mere information omkring hvordan Remote Validation virker, kan findes i dokumentationen under afsnit 7.3 Remote Validation.

## 7.4 Favorites

I denne iteration blev konceptet favoritter implementeret. Efter at systemet udvikler sig til at indeholde mere og der samtidig blev oprettet flere og flere quizzes (og nu grupper) manglede der en måde at holde styr på de quizzes og grupper, som brugeren ikke selv har oprettet, men gerne vil have hurtig adgang til. For at løse dette blev der oprettet to nye entiteter til databasen: **FavoriteGroups** og **FavoriteQuizzes**. Disse sørger for at håndtere mapningen mellem brugere og deres favoritter. Favoritterne ses i en tabel, hvor der både er mulighed for at fjerne sin favorit, men også gå til en gruppe eller besvare en quiz.

Favoritter kan tilføjes rundt på hjemmesiden ved at trykke på en stjerne ud for et søgeresultat eller når man er i gang med at besvare en quiz. Det håndteres med AJAX, så siden ikke genindlæses når man trykker på dem.

## 7.5 Azure web- og database-server

I iteration 2 fandt vi ud af at Facebook-login muligheden ikke var funktionel når database- og webserveren var på ASE's interne netværk. Derfor skiftede vi til en Microsoft Azure server [Microsoft, 2015b]. I samme omgang blev databaseserveren udskiftet med en Microsoft SQL Server [Microsoft, 2015d] også tilgængelig igennem Azure.

Fordele:

- Brugeren udenfor det interne netværk kan tilgå webapplikationen
- Højere hastighed og dermed bedre brugeroplevelse
- Facebook-login virker efter hensigten

Ulemper:

- Ikke gratis

Fordele opvejer klart ulemperne. At brugere udefra kan tilgå hjemmesiden er selvfølgelig fundamental for projektet og vægter derfor højest.

## 7.6 Sprints

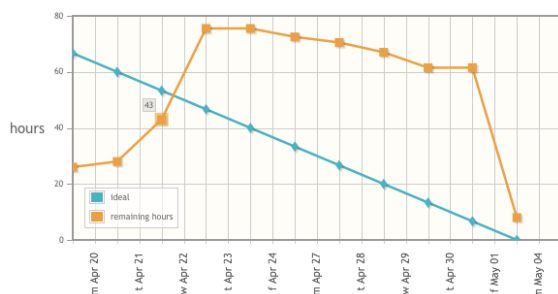
I denne iteration var der to sprints. De var begge to uger. Nøgletallene for iterationen er anført i tabel 7.1

De forventede timer i hvert sprint er højere end sidst. Dette skyldes, at vi gerne ville nå at have de sidste essentielle funktioner med inden projektet afsluttes. Umiddelbart er resultatet for timerne for sprint 1 gode, men igen er tallene ikke retvisende i sig selv. Sprintet havde en lignende start som de tre i iteration 2. Vi nåede at lave opgaverne til

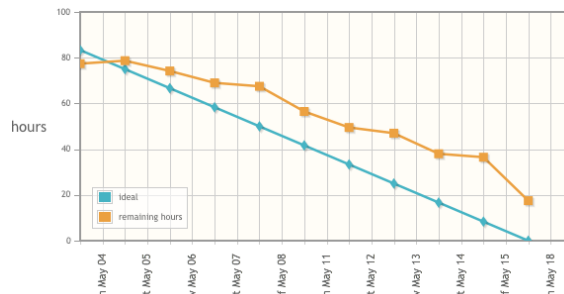
Sprint	Opgaver	Forventede timer	Registrerede timer	Dækning
1	47	70	75	107 %
2	79	93	144	154 %

**Tabel 7.1.** Tabel med nøgletal fra iteration 3

vores opstartsmøde, men fik ikke estimeret nogle timer. Burndown-grafen, på figur 7.3, er derfor tæt på intetsigende. Syv opgaver blev flyttet til næste sprint fordi de ikke nåede i mål. Grafen for sprint 2, figur 7.4 er væsentligt pænere. Det viser at vi har arbejdet jævnt i løbet af de to uger og næsten er nået i mål. Timeforbruget er dog væsentligt over hvad vi havde forventet. Vi har brugt 51 timer mere end forventet på de to uger.



**Figur 7.3.** Burndown-graf for sprint 1

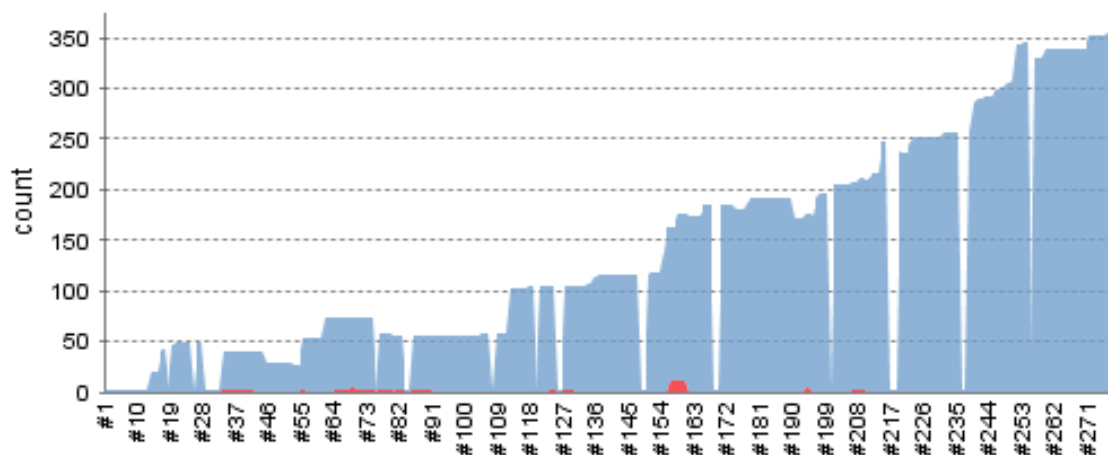


**Figur 7.4.** Burndown-graf for sprint 2

## 7.7 Test

Denne iteration bød på en del ændringer af eksisterende funktionalitet i stedet for nye funktioner. Her kommer de oprettede tests rigtigt i spil, da ændringerne hele tiden kan testes for om de ødelægger noget af den funktionalitet som var der i forvejen.

På figur 7.5 er et udklip af de udførte tests for starten af projektet og frem til slutningen af iteration 3.



**Figur 7.5.** Automatisk udførte tests på Jenkins-server for iteration 1, 2 og 3

Der har været en rimelig udvikling af tests under hele projektet. Dog skulle kurven, ideelt set, gerne have været mere lineær fra start til slut. Integrationstestene er også lavet i denne iteration. De er lavet ud fra Collaboration-metoden og tester afhængigheder i mellem modulerne ud fra deres logiske sammenhæng i forbindelse med den enkelte use cases.

Ved afslutningen af iterationen har vi udført de accepttests der er lavet sideløbende med use case-beskrivelserne. De er alle godkendt.



# 8 Projektforløb og opfølgning

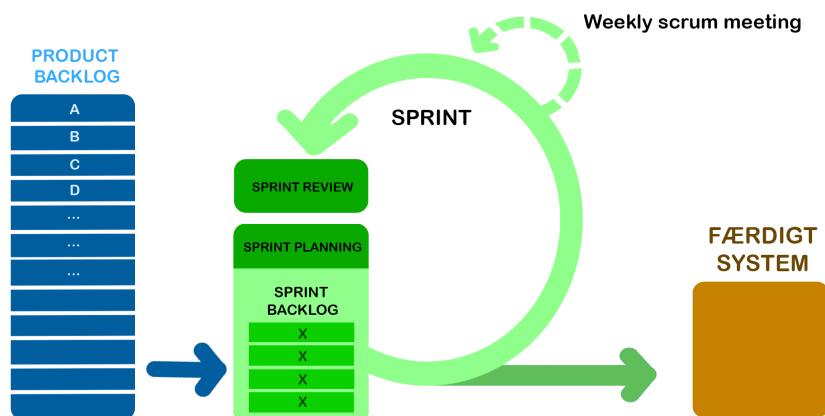
Her følger beskrivelser af arbejdsprocessen, resultater og erfaringer samt beskrivelser af vores udviklingsværktøjer og konklusioner for projektet.

## 8.1 Arbejdsproces

### 8.1.1 Udviklingsmodeller

Projektgennemførelsen er foregået indenfor agile rammer, da gruppens medlemmer fra foregående semestre har opnået erfaring med denne arbejdsmetode. Det var klart fra starten at der unægteligt ville forekomme ændringer i systemet, bl.a. fordi en del af de værktøjer, der skulle benyttes først ville blive demonstreret i kurser senere på semesteret. Bl.a. kan Microsofts Entity Framework [Microsoft, 2015a] nævnes. Dette lærte vi først om i I4DAB efter at den første instans af databasen og tilhørende CRUD metoder var oprettet. Entity Framework ville dog simplificere opgaven, og det stod klart at den burde erstatte den daværende database access metode, som var oprettet. Af denne grund var det derfor oplagt arbejde agilt i modsætning til f.eks. en mindre fleksibel form som vandfaldsmodellen således at vi ikke ville blive påvirket af uundgåelige ændringer.

I forlængelse af de agile rammer blev det valgt arbejde iterativt med projektet - nærmere bestemt valgte vi at tage udgangspunkt i SCRUM [MountainGoatSoftware, 2015]. SCRUMs værktøjer har givet gruppen nogle faste regler at følge for at nå i mål med projektet, og det har bl.a. betydet at arbejdet er skredet frem i ugelange sprints og hver gang en ny "milepæl" i projektet er nået er projektet rykket ind i en ny iteration. Figur 8.1 viser en grov skabelon over hvordan vi har udnyttet Scrums egenskaber.



*Figur 8.1.* Skabelon, der illustrerer hvordan vi har benyttet scrum

I starten af projektet blev systemets use cases udarbejdet og disse use cases har udgjort den egentlige product backlog for projektet. Ved begyndelse af en ny iteration har gruppen vedtaget hvilke use cases, der har skullet implementeres i det kommende sprint. Gruppen har mødtes fast to gange om ugen, men derudover har vi haft forelæsninger sammen, og der er i denne sammenhæng opstået mere uformelle møder hvor eventuelle problemer eller nye forslag er blevet drøftet igennem. Efter hvert endt sprint er der blevet holdt sprintreview, hvor sprintet er blevet evalueret så næste sprint har kunnet gøres endnu bedre.

Selvom en masse af SCRUMs værktøjer er blevet benyttet er der stadig visse elementer der bevidst er udeladt, bl.a. har der hverken været en product owner eller scrum master. Eftersom at produktets krav ikke er styret af en kunde, men i stedet af hele gruppen, har det ikke været relevant at have en product owner.

Igennem projektet har gruppen udviklet sig og er på visse punkter blevet bedre til at strukturere arbejdet. Gruppen har fra starten af udvist disciplin og arbejdet for at færdiggøre backloggen for et bestemt sprint. I starten forgik struktureringen ved at backloggen for et nyt sprint blev udarbejdet i starten af sprintet og medlemmerne kunne herefter selv vælge de tasks som de ville udføre. Denne måde at uddele arbejdet på (eller mangel på samme!) gjorde at visse tasks ikke blev taget før sent i projektet og senere begyndte hvert sprint derfor med at gruppen sammen uddelte opgaverne. Dette havde samtidig den effekt at taskene blev bedre koordineret således at tasks inden for et bestemt område gik til samme person.

Arbejdsfordelingen er sket ud fra de enkelte gruppemedlemmers interesser, og der er ikke blevet uddelt bestemte områder til medlemmerne. Det har derfor ofte været tilfældet at et medlem der f.eks. har stået for database-delen i ét sprint, har implementeret en controller eller view i et andet. Denne måde at fordele arbejdet på er ikke blevet gjort fordi det er den optimale måde at arbejde produktivt (tværtimod havde faste arbejdsområder måske været mere logisk), men derimod ud fra et ønske fra gruppens medlemmer om at opnå bedst mulig erfaring med de forskellige teknologier som projektet har budt på.

### 8.1.2 Møder og referater

Så vidt det har været muligt er der blevet afholdt to faste møder om ugen i projektforløbet. Det ene møde var et ugentligt statusmøde hvor gruppemedlemmerne fortalte hvad de har lavet siden sidste møde og hvilke udfordringer de står overfor. I forbindelse med dette møde blev der også ved hver sprintafslutning planlagt nyt sprint, og de tasks der ikke blev færdige kunne komme med videre i det nye sprint. Ligeledes har der også været forsøgt at holde møde med vejleder ugentligt, når der har været problemstillinger, som har været nødvendige at få afklaret, eller der har været en ny version af produktet som kunne vises frem.

Der er også blevet skrevet referater ved næsten møderne for at dokumentere de emner og problemer som er taget op og afklaret. Disse er vedlagt på bilags-CDen. Se afsnit 10 for detaljer. Enkelte møder er dog ikke dokumenteret, hvilket skyldes at mødet er afholdt som et stand-up møde.

### 8.1.3 Arbejdsfordeling

I starten af projektførløbet blev det aftalt at man selv skulle sørge for at tage opgaver på taskboardet på Redmine, men da det ikke blev anvendt som forventet, gik vi over til at uddele opgaver ved hver sprintstart. Dette gjorde vi af flere årsager, hvoraf den ene var at vi ville forsøge at skabe en mere ligelig arbejdsfordeling af opgaver og gøre det mere overskueligt at se, hvem der arbejdede på hvilke opgaver. Til de store beslutninger i opstartsfasen har gruppen siddet sammen og fået afklaret de store linjer, ellers har gruppens medlemmer arbejdet meget selvstændigt i gennem forløbet, hvor de ugentlige møder har afklaret opgaver og udfordringer og taskboardet på Redmine har været samlingspunktet for arbejdsfordelingen.

### Rollefordeling

Enkelte gruppemedlemmer har haft en overordnet rolle i løbet af projektet. Bjørn har været ansvarlig for L<sup>A</sup>T<sub>E</sub>X - opsætning.

## 8.2 Udviklingsværktøjer

Her følger en beskrivelse af de anvendte udviklingsværktøjer i projektet.

### IDE

- Microsoft Visual Studio 2013

Microsoft Visual Studio har været et oplagt valg at benytte som IDE, da det er dette, der er blevet benyttet på samtlige tidligere semestre og det er da også dette vi har benyttet på nuværende semesters kurser. Desuden var det et naturligt valg, da vi besluttede at benytte webmiljøet ASP.NET MVC [Microsoft, 2014] til at udvikle applikationen i.

### Frameworks

- Microsoft ASP.NET MVC
- Microsoft Identity
- Microsoft Entity Framework

MVC modellen har for alvor vist sin fulde styrke gennem dette semester. Model-view-controller arkitekturen har gjort det muligt at arbejde og uddele opgaverne på tværs af gruppen således at nogle stod for front end mens andre stod for back end uden at træde hinanden over fødderne. Identity-frameworket har stået for brugeradministrationen og er integreret tæt sammen med MVC-strukturen. Entity Framework blev først benyttet ca. halvvejs gennem projektet, da det først var her vi fik oplæring i det. Dette ORM-framework har hjulpet gevaldigt og speedet udviklingen af databasens tabeller en hel del op.

### Versionsstyring og Continuous Intergration

- Git

- Jenkins

Uden et versionstyringsværktøj ville det have været tæt på umuligt at arbejde så mange på projektet uden at have overskrevet ændringer fra hinanden. Desuden har vi anvendt Jenkins [Kawaguchi, 2015] som CI-server. Dette har fungeret godt og givet et godt overblik over kodebasens udvikling.

## Arbejdsproces

- Scrum

Scrum har alt i alt fungeret godt for gruppen, og det har bl.a. hjulpet os med at strukturere arbejdet med projektet, ved at have givet et værktøj til at inddele arbejdet i sprints af 1-3 uger.

## Diverse

- Redmine
- Microsoft Visio
- L<sup>A</sup>T<sub>E</sub>X
- DDS-lite

Redmine er blevet brugt som gruppens ”scrum-forum” og det er her de enkelte sprints er blevet oprettet og tasks uddelegeret til gruppens medlemmer.

Gruppen vidste fra starten at der skulle oprettes mange modeller og figurer til projektet. I begyndelsen blev StarUML valgt, men da det ikke levede op til gruppens behov blev Microsoft Visio valgt i stedet.

Rapporten og dokumentationen er skrevet i L<sup>A</sup>T<sub>E</sub>X som er et kodebaseret tekstredigeringsværktøj. Dette er valgt fordi den kodebaserede måde at skrive på, sikrer god integration med Git-styring.

## 8.3 Fremtidigt arbejde

Arbejdet med dette semesterprojekt er færdiggjort, men Quiz Creator kan udvikle sig meget mere. Mange funktionaliteter kan tilføjes. Blandt andet kan nævnes:

### Private indstillinger

Hvis en gruppe eller quiz kun er tiltænkt bestemte brugere.

### Konfiguration af quizzer og grupper

Information om quizzens formål, begrænsning af antallet af gange en bruger kan besvare en quiz, tidsbegrænsning på quizzen eller de enkelte spørgsmål.

### Præsentation af resultater

Præsentation af resultater undervejs, eller slet ikke hvis det er en eksamensopgave.



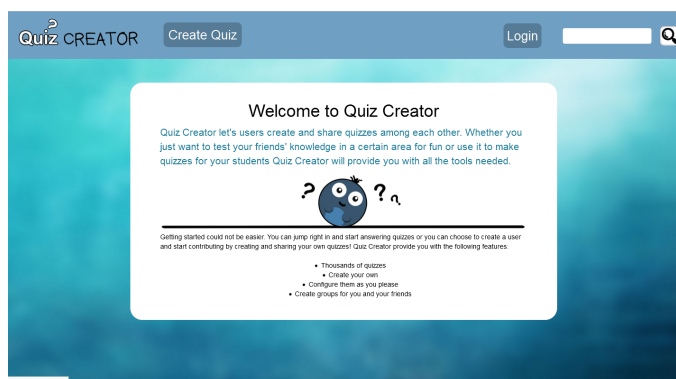
Selvom gruppen generelt er tilfreds med det grafiske indtryk af GUIet, så må det erkendes at det ikke er her de fleste kræfter er blevet lagt, og det vil derfor også være en oplagt mulighed at justere og forbedre designet i kommende versioner.

Dette er alt sammen smårettelser som samlet er med til at give brugeren den bedst-mulige oplevelse, men den om nok vigtigste feature som med rette kunne implementeres i en kommende version er muligheden for at afvikle applikationen på mobile-enheder såsom iOS- og Android-enheder. Dette vil passe godt med det overordnede formål med Quiz Creator og det ville betyde at brugerne ville kunne besvare quizzes i hjemmet, på arbejdspladsen, i skolen, men nu også på farten.

Hvorom alt er, så er Quiz Creator i sit nuværende stadie en funktionel og brugervenlig applikation, men der er ingen tvivl om at en masse features kan inkluderes endnu.

## 8.4 Resultater

Dette afsnit redegør for det endelige resultat af projektet efter arbejdet med det er endt. Projektets endelige resultat vurderes på baggrund af de accepttests som er blevet opstillet både for use cases og ikke-funktionelle krav. Efter accepttestene er foretaget står det klart at Quiz Creator opfylder de fleste krav som er sat til projektet. Dette afsnit vil give et kort indblik i hvordan scenariet for use casen Answer Quiz fungerer. Når en bruger besøger Quiz Creator vil man blive mødt af en velkomst skærm, som er vist på figur 8.2



**Figur 8.2.** Forside på Quiz Creator

Herfra kan brugeren vælge at søge efter en quiz, logge ind og hvis man er logget ind kan man desuden oprette en quiz, se sine grupper eller quizzes. Hvis brugeren vælger at søge efter en quiz vil de quizzes der matcher søgeordene (dette gælder både tags som er tildelt quizen og quizzens navn), blive vist for brugeren. Quizzene opfylder de ikke-funktionelle krav og vil bestå af minimum to spørgsmål. Når en quiz vælges vil brugeren blive præsenteret for det første spørgsmål, og man har nu mulighed for at besvare dette og dernæst gå videre til næste spørgsmål eller gå til et vilkårligt spørgsmål i quizen via quickaccess baren over spørgsmålet. Når alle spørgsmål er besvaret vil man få mulighed for at trykke på en knap for at afslutte quizen. Når man gør dette vises en skærm, der afslører resultaterne for den netop besvarede quiz.

Alt i alt er projektet implementeret tilfredsstillende i forhold til gruppens ambitioner.

Produktet kan ses og ”leges med” ved at benytte følgende URL og logge ind med oplysningerne:

URL: `http://quizcreator.azurewebsites.net`

Email: `testmail@test.dk`

Password: `password`

På siden har brugeren oprettet en testgruppe og testquiz. Der kan findes yderligere quizzes ved at søge på tagget ”iha”.

## 8.5 Erfaringer

Arbejdet med dette semesterprojekt har givet en masse erfaring til gruppens medlemmer. Der er både opnået erfaring med rent tekniske emner, men der er også opnået bedre forståelse for fx arbejdsmetoder som scrum og strukturering af et projekt i dette omfang.

Teknisk er der opnået god forståelse indenfor en lang række af emner. Først og fremmest er der opnået en god forståelse for, hvordan man inkorporerer og benytter en MSSQL database i sit projekt og ligeledes hvordan man udfører CRUD operationer på denne, både ved brug af ADO.NET og det mere gnidningsfri Microsoft entity framework. Desuden har gruppen fået erfaring i at udvikle web-applikationer med ASP.NET frameworket som både har givet erfaring med client- men også server side programmering. Eftersom det er en web-applikation har det ikke kunne undgås ikke at få en bedre forståelse for markup sproget HTML, CSS og programmeringssproget JavaScript. Dette har også gjort at vi er blevet introduceret til jQuery-biblioteket og desuden også asynkrone request-kald til serveren med AJAX-teknologien.

MVC strukturen i projektet har givet en god forståelse for denne softwarearkitektur, og det har især vist sit værd når opgaver skulle uddelegeres, hvor en person kunne stå for design af et view, mens en anden tog sig af controlleren og en tredje arbejdede med datatilgangen i DAL laget.

Dette semesterprojekt er det første af vores semesterprojekter som har budt på et rent softwareprojekt, og det har været et krav at arbejdet har skullet foregå agilt. Gruppen valgte derfor at udnytte en del af værktøjerne fra Scrum. Det viste sig hurtigt at dette satte nogle gode stabile rammer for projektet: sprints har som regel taget i omegnen af 1-2 uger og det har gjort at vi har haft en konsistent arbejdsrytme og har sat nogle naturlige mållinjer for hvornår bestemte features har skulle være overstået. Desuden har vi fået fornuftig indlæring i, hvordan man for fulde udnytter et agilt arbejdsmiljø. Igennem iterationer er eventuelle ændringer og rettelser ivrigt blevet diskuteret og hvis noget har skulle refaktureres har det sjældent været noget større problem fordi arbejdet har været så iterations-fokuseret - som eksempel kan nævnes refakturering af DAL-laget: i starten blev databasen tilgået med ADO.NET teknologien men efter vi i DAB-kurset blev introduceret til Microsofts Entity Framework blev dette hurtigt en erstatning for ADO.NET. Sådanne relativt store ændringer i systemet ville unægteligt have været mere besværligt i en arbejdsmetode som vandfaldsmodellen.

På et mere formelt men bestemt ligeså brugbart plan har vi fået en masse erfaring med at arbejde på samme projekt i et større team med versionsstyringsværktøjet Git. Git har gjort det muligt for os at arbejde på samme Visual Studio projekt ved at merge ændrede

filer sammen.

Alt i alt har arbejdet med dette semesterprojekt givet en god erfaring med en masse forskellige aspekter, og det har gået fornuftigt hånd i hånd med de resterende 4. semesterskurser og det har suppleret hinanden begge veje således at det vi har lært i undervisningen har kunnet bruges i projektet og omvendt. Erfaringerne som er blevet gjort vil uden tvivl gavne samtlige medlemmer i fremtiden, hvad end det er under bacheloren eller på arbejdspladsen.



Her følger samlede og individuelle konklusioner om arbejdet med projektet.

## 9.1 Samlet konklusion

Projektforløbet for dette 4. semesterprojekt har forløbet over fire måneder sideløbende med den almindelige undervisning. Gruppen har lagt mange timer i projektet og det færdige resultat opfylder både gruppens, såvel som ASEs krav til applikationen.

Den nødvendige funktionalitet, som foreskrevet af use casene, er implementeret. Kort ridset op betyder det at Quiz Creator er en webapplikation, som lader en bruger logge ind, oprette quizzes, arrangere disse i grupper, besvare andre quizzes og udpege de bedste quizzes igennem et favorit-system.

Alt i alt er gruppen tilfreds med applikationen og den lærdom, som er blevet tilegnet gennem forløbet har gået godt i spænd med det materiale som 4. semesters-fagene har budt på. Ydermere er gruppen gennem arbejdet med Scrum-frameworket gradvist blevet bedre og bedre til at estimere den nødvendige tid de givne opgaver har taget - dette kommer bl.a. til udtryk i burndown diagrammerne som har indgået i hvert sprint.

## 9.2 Individuelle konklusioner

### Anders Bæk Møller

Arbejdet med dette semesterprojekt har været enormt lærerigt. Mine primære ansvarsområder i projektet har været use casene Answer Quiz og Search, og arbejdet med dette har givet mig en god forståelse for hvordan web applikationer skal opbygges, og helt konkret har jeg fået indlæring i brugen af HTML, CSS, Javascript, JQuery-biblioteket og AJAX-teknologien.

På det arbejdsmæssige plan har jeg opnået en masse brugbar erfaring til senere projekter: 4. semesterprojekt har budt på det første rene software projekt, og det har betydet, at vi har taget de agile arbejdsprincipper i brug, hvilket har gjort at ændringer ikke har været opfattet som en belastning, men derimod som en naturlig udvikling i projektet. En anden vigtig erfaring fra projektet er læren om software arkitektur. Vi har benyttet MVC-modellen i ASP.net og det er en arkitektur, der for alvor har vist sit værd eftersom det har fjernet koplingen mellem frontend og backend-delen. Arbejdet med projektet har også lært mig en del, omend det på nogle punkter efter min mening ikke altid har fungeret optimalt. Vi har benyttet en del af scrums værktøjer og det har helt sikkert lært mig en del om at sætte mål for projektet og opdele arbejdet i iterationer. Vi har ikke altid været lige gode til

at bedømme arbejdsbyrden i starten af et sprint, men ikke desto mindre er jeg stadig blevet bedre til at estimere tidsforbruget for en given task. Gruppens medlemmer har arbejdet meget separat, og her kunne jeg til tider godt have ønsket at der havde været flere dage, hvor vi havde siddet og arbejdet sammen på holdet. En anden ting jeg har savnet er at hver medlem har haft et fast arbejdsområde, istedet for at medlemmer har "hoppet" rundt på tasks. Dette kunne formentlig have givet en mere konsistent rollefordeling og det havde uden tvivl været nemmere at uddelegere tasks.

Ser man bort fra dette har jeg alt i fået meget brugbar erfaring ud af dette projekt, og det er en erfaring som jeg vil kunne bruge i mine projekter i fremtiden.

### **Bjørn Sørensen**

Dette semesters projekt har været et meget anderledes forløb sammenlignet med de tidligere semesterprojekter. Tidligere har vi været to programmører i gruppen, som har haft det samlede ansvar for struktureringen af alt softwaren. I år har vi været en gruppe af syv, med forskellige baggrunde og forskellige tilgange.

Opstarten af projektet tegnede lovende og vi fik snakket forskellige idéer godt igennem. Grundidéen vi endte med har været et spændende udgangspunkt og med rige muligheder for udvidelser. Desværre viste det sig at vi nåede væsentligt mindre end jeg havde forventet. Dette tror jeg skyldes at vi som gruppe ikke har ydet det nødvendige hen over semesteret. Der ud over har vi bøvlet med simple ting som at overholde deadlines og være aktive og målrettede til møder samt at prioritere sine timer til projektet.

Jeg har savnet at der har været en hård styringen af gruppens indsats. Scrum-elementerne vi har anvendt har givet gode overblik, men har også haft den konsekvens at arbejdsfordelingen har været skæv, gruppemedlemmerne i mellem. Jeg har dog valgt at prøve at fokusere energien på at komme frem i projektet og ikke tænke så meget på hvor meget andre ligger i det.

Set over hele semesteret har jeg dog stadig fået et stort udbytte af projektet og lært meget om web-udvikling og struktureringen af denne.

### **Jesper Christensen**

De sidste to semesterprojekter har jeg været i en projekt gruppe kun bestående af to IKT studerende, mig selv inklusiv. Det har betydet, at vi kun var to til at lave software-arkitekturen og implementere det. Dette semester bestod projektgruppen af syv IKT-studerende, hvilket har betydet at der har været meget mere software, som skulle koordineres, og mange flere meninger samt forskellige tilgange til tingene.

Udbyttet fra 4. semesterprojekt har været rigtig stort. Det er første gang jeg arbejder med web-teknologi, så alt inden for området har været nyt for mig, lige fra HTML til jQuery og JavaScript. Vi har haft mange teknologier og frameworks involveret i dette projekt, og de har alle været spændende at lære og arbejde med. Jeg går meget op i at udvikle mig og lære så meget jeg kan. Derfor er jeg også tilfreds med den viden og erfaring, som jeg har tilegnet mig i dette projekt.

I forhold til forrige projekter har tilgangen og arkitekturen været meget anderledes. Vi har benyttet Scrum-elementer og agil udvikling, hvilket jeg synes har været rigtig spændende. Dog har der været ting, som ikke altid fungerede for os. Arbejdsfordelingen har været skæv, da det ikke har været alle, som var lige gode til selv at have ansvaret for at tage opgaver, i stedet for at få dem givet. Derudover har vi oplevet, at det var meget svært at estimere tid på de forskellige opgaver, specielt når vi skulle bruge teknologier, som vi ikke havde arbejdet med før. Undervisningen til web-applikationer lå meget sent på semesteret, og vi har derfor skulle lære mange af tingene selv, da vi endnu ikke havde haft undervisning i det.

Jeg er meget tilfreds med det faglige udbytte jeg har fået af forløbet. Produktet synes jeg ikke helt har ramt det ambitions-niveau vi havde sat fra starten og det er selvfølgelig ærgerligt. Jeg synes at størstedelen af projektgruppen har arbejdet meget og målrettet for at opnå den funktionalitet vi har. Der er dog plads til flere forbedringer og fremtidigt arbejde.

### **Lars Harup Holm**

Dette semesterprojekt er det første af slagsen hvor IKT/E/EP er delt op. Det har været en spændende overgang, hvor hele projektgennemførslen er blevet omstruktureret. Dette har dog heller ikke været uden sine udfordringer. Vi har i gruppen været 7 mand, hvilket nok har været et par for mange til at opnå den helt rigtige gruppesynergi ved dette specifikke projekt, når alle 7 medlemmer arbejder med software.

Valget på at arbejde med web-udvikling har været en stor mundfuld for mig, da jeg ingen tidligere erfaring har haft med web, og GUI-kursets forelæsning om emnet lå sidst på semesteret. Jeg synes dog, trods en langsom opstart, at mit udbytte af arbejdet er enormt, hvilket jeg til dels kan takke enkelte gruppe-medlemmers overskud til at dele deres forudliggende viden, når jeg har haft problemer.

Arbejdsprocessen som en iterativ udvikling har været en anderledes, men interessant omvæltning. Denne kombineret med elementer fra SCRUMs agile tilgange har været en særdeles lærerig proces både på godt og ondt. Da der i gruppen har været en meget skæv arbejdsfordeling har vi igennem projektets forløb ændret tilgangen til SCRUMs taskboard og sprint elementer, så disse blev tilrettelagt bedre til gruppens arbejdsform.

Alt i alt har jeg dog flyttet mig meget, både fagligt og personligt, i løbet af projektet. Jeg kan tage de gode og dårlige oplevelser med som erfaring om både softwareudvikling men i sandhed også om arbejdsprocesser.

### **Lasse Beck Thostrup**

Dette semesterprojekt har været noget helt nyt, fordi der nu var krav om at vi skulle arbejde iterativt, og at vi ikke længere skulle være sammen med elektroingeniørerne. Sidstnævnte har betydet at vi skulle være bedre til at koordinere arbejdet for syv personer, der alle arbejder i det samme Visual Studio projekt. Her har det været rigtig brugbart at bruge SCRUM-taskboardet, hvormed det har været nemt at se hvad forskellige personer arbejder med. Desværre var alle ikke lige så gode til at bruge boardet, og det medførte at der ikke altid blev logget tid, så burndown-chart mm. begyndte at skride. Hvis man skal gøre brug

af disse værktøjer til projektstyringen, skal alle engagere sig 100%, ellers falder det nemt til jorden. Heldigvis blev vi alle mere vant til at bruge disse værktøjer, og det begyndte at give god mening og effekt.

Jeg synes, at vi har formået at få skabt et godt produkt inden for et teknologi-område, som få af os har arbejdet med før. Det har været enormt lærerigt både inden for web-teknologi, men også inden for arbejdsprocesser. Det har været fedt, at vi har kunne rotere mellem arbejdsområder, sådan at vi hver især har kunne få erfaringer med de forskellige dele, så som database, model, view og controller.

Alt i alt har det været et rigtig godt semesterprojekt med stort udbytte.

### **Loc Dai Le**

Dette semesterprojekt har været spændende og lærerigt, da det har givet mig muligheden til at anvende min teoretiske viden fra de forskellige kurser på 4. semester til noget praktisk. Derudover har det givet mig en bredere forståelse for, hvordan det er at arbejde i et større team til at udvikle en specifik IT-løsning. Igen har vores projektgennemførelse været baseret på Scrum-princippet. Da Scrum er en meget stor og kompliceret agil udviklingsmetode, var det lidt langhåret at anvende Scrum fuldt ud i starten. Men eftersom vi fik vores Backlog og Taskboard op at køre, blev Scrum en del af arbejdsrutinen. Ved brug af Scrum har vi alle været på lige fod med projektet, og vores arbejde har været struktureret og velgennemført.

Udover Scrum blev projektet også gennemført ved brug af iterationer. Det er første gang jeg har beskæftiget mig med iterationer når det gælder projektgennemførelse. Ved at arbejde iterativt har vi opfyldt de vigtigste krav for vores system sekventiel.

I dette semesterprojekt har vi arbejdet med webudvikling. Vi har anvendt ASP.NET MVC-frameworket til at udvikle vores web-applikation. Som projektet skrider frem, har jeg fået en bredere viden inden for webudvikling, og de teknologier der anvendes i sammenhæng med webprogrammering. Jeg har i dette projekt beskæftiget mig med teknologier som HTML5, CSS3, JavaScript, AJAX, jQuery, håndtering af databaser med Entity Framework og andre .NET frameworks.

I gruppen har vi alle været gode til at hjælpe hinanden, og kommunikationen mellem gruppens medlemmer har været på et tilfredsstillende niveau. Der har været episoder, hvor vi har været uenig om nogen ting, men det blev altid løst på en fornuftig måde.

### **Michael Toft Jensen**

Dette semesterprojekt har været en del anderledes end tidligere projekter, da der i tidligere projekter har været involveret elektronik. På dette semester har vi været syv mand på et softwareprojekt. Det er mange personer som man skal forsøge at holde involveret i et projekt, især når vi kommer med forskellig baggrund og forskellige arbejdsmetoder, samt de fleste af os ikke har arbejdet sammen tidligere. I starten virkede det lovende og der var bred enighed og samling om tingene, samt gruppen fik lagt en masse ideer på bordet. Jeg tror der er enighed om at vi ikke fik nået alt det vi ville nå, hvilket kan skyldes at der ikke er lagt de timer i arbejdet der skal, eller at måden vi har arbejdet på ikke har været



optimal. Udover har der, som semesteret skred frem, været en skæv arbejdsfordeling, og der har været en lille mangel på struktur.

Jeg har savnet at der har været en bedre koordinering i gruppen og en bedre uddelegering af opgaver, og at vi som gruppe havde siddet mere sammen og arbejdet, så man kunne få mere vidensdeling og sparring ud af hinanden. Nogle af de Scrum-elementer vi har brugt har fungeret godt, mens andre ikke er blevet anvendt som forventet og har derfor været mindre anvendelige og værdiskabende.

Jeg er blevet udfordret og har fået en masse ud af dette projekt, og lært en masse om web-udvikling både back-end, front-end, hvordan man opbygger en webapplikation og hvilke udfordringer der ligger heri.



På bilags-cden vil det være muligt at finde doxygen tilhørende vores kodebase, selve kodebasen, en pdf udgave af rapport og af dokumentationen, samt mødereferater. Doxygen er som beskrevet tidligere en række HTML filer, som vil være at finde i den respektive mappe. De andre dele vil også være at finde i respektive mapper, så det er nemt at navigere og finde de forskellige dele. På cd'en vil der i mappen "HtmlClassModel" kunne findes en Html-dokumentation for klassediagrammer i systemet.



# Litteratur

---

- Dimitri van Heesch. Doxygen, 2015. URL <http://www.stack.nl/~dimitri/doxygen/index.html>. Accessed 22.05.2015.
- Kohsuke Kawaguchi. Jenkins CI, 2015. URL <https://jenkins-ci.org>. Accessed 24.05.2015.
- Microsoft. ASP.NET MVC, 2014. URL <http://www.asp.net/mvc>. Accessed 24.05.2015.
- Microsoft. Entity Framework, 2015a. URL <https://msdn.microsoft.com/da-dk/data/ef>. Accessed 25.05.2015.
- Microsoft. Azure, 2015b. URL <http://azure.microsoft.com/>. Accessed 24.05.2015.
- Microsoft. Identity MSDN, 2015c. URL <https://msdn.microsoft.com/en-us/library/dn613290%28v=vs.108%29.aspx>. Accessed 25.05.2015.
- Microsoft. SQL Server, 2015d. URL <http://www.microsoft.com/SQLServer>. Accessed 24.05.2015.
- MountainGoatSoftware. Scrum Methodology and Project Management, 2015. URL <https://www.mountaingoatsoftware.com/agile/scrum>. Accessed 25.05.2015.
- Olivier Müller, Marc Delisle, and Loïc Chapeaux. phpMyAdmin, 2015. URL <http://www.phpmyadmin.net/>. Accessed 24.05.2015.
- Charlie Poole, Rob Prouse, and Simone Busoli. NUnit, 2014. URL <http://www.nunit.org>. Accessed 24.05.2015.